CMP
United Business Media

PRODUCT REVIEWS: CODEWARRIOR 9 PALM OS, CHARACTER STUDIO 4

APRIL 2003

GameDeveloper

ON THE FRONT LINE OF GAM

HAVE A HEART

THE EMOTIONAL HEART OF ART DIRECTION

THE PLAY'S THE THING

FPS CONSOLE CONVERSION

Postmortem: Lost Toys' BATTLE ENGINE AQUILA

GameDeveloper **Mobile** INSIDE!

$5.95US $6.95CAN
04>
0 71486 02133 9
Display Until April 25, 2003

CONTENTS

VOLUME 10, NUMBER 4

April 2003

FEATURES

DEPARTMENTS

COLUMNS

COVER: *3D design of the Battle Engine Aquila and its driver, Hawk Winter. Image created by the art team at Lost Toys.*

Game Developer (ISSN 1073-922X) is published monthly by CMP Media LLC, 600 Harrison St., San Francisco, CA 94107, (415) 947-6000. Please direct advertising and editorial inquiries to this address. Canadian Registered for GST as CMP Media LLC, GST No. R13288078, Customer No. 2116057, Agreement No. 40011901. **SUBSCRIPTION RATES:** Subscription rate for the U.S. is $49.95 for twelve issues. Countries outside the U.S. must be prepaid in U.S. funds drawn on a U.S. bank or via credit card. Canada/Mexico: $69.95; all other countries: $99.95 (issues shipped via air delivery). Periodical postage paid at San Francisco, CA and additional mailing offices. **POSTMASTER:** Send address changes to Game Developer, P.O. Box 1274, Skokie, IL 60076-8274. **CUSTOMER SERVICE:** For subscription orders and changes of address, call toll-free in the U.S. (800) 250-2429 or fax (847) 647-5972. All other countries call (1) (847) 647-5928 or fax (1) (847) 647-5972. Send payments to *Game Developer*, P.O. Box 1274, Skokie, IL 60076-8274. For back issues write to *Game Developer*, 1601 W. 23rd St. Ste. 200, Lawrence, KS 66046-2703. Call toll-free in the U.S./Canada (800) 444-4881 or fax (785) 841-2624. All other countries call (1) (785) 841-1631 or fax (1) (785) 841-2624. Please remember to indicate *Game Developer* on any correspondence.

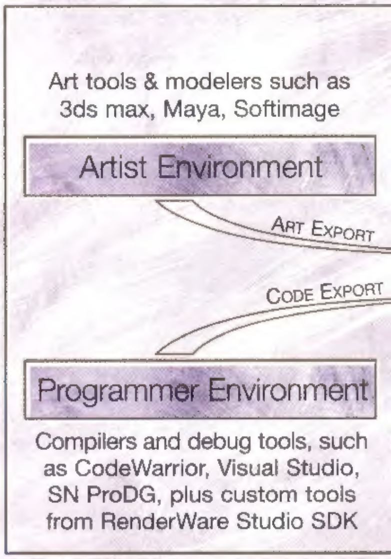
RenderWare

RenderWare Studio

RenderWare Studio is a unique collaborative game production system that allows the whole team to create, view and tune games in real-time on multiple platforms.

RenderWare Studio encompasses the entire development process from pre-production to QA, allowing all team members to work in parallel and remove time-consuming bottlenecks.

PC Host



Art tools & modelers such as 3ds max, Maya, Softimage

Compilers and debug tools, such as CodeWarrior, Visual Studio, SN ProDG, plus custom tools from RenderWare Studio SDK

RenderWare Platform

RenderWare Platform is the most comprehensive suite of development tools and technologies available, seamlessly integrating Graphics, Audio, AI and Physics.

The industry standard in middleware is now more powerful than ever before, offering unrivalled performance – *more than twice the power of other graphics solutions.*

Target

Graphics

- High performance, low footprint, easily customized libraries with PowerPipe™ API
- Comprehensive art-path tool chain
- Collision detection
- Cutting edge, genre specific FX Packs
- Target-specific enhancements
- VisionFX advanced occlusion plug-in

We are proud to be sponsors of...

igda international game developers association

game developers choice awards

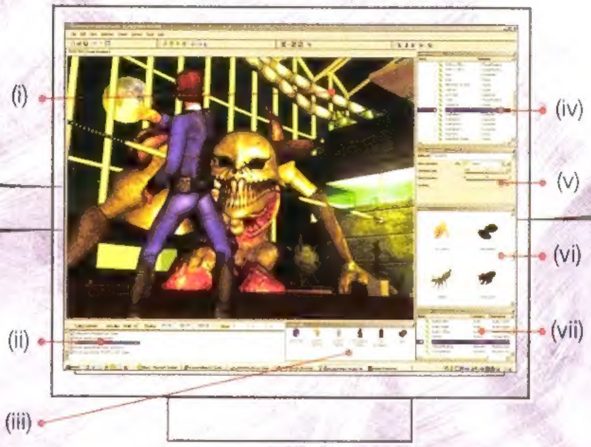
For further information or a personal demonstration of RenderWare, please email rw-info@csl.com, or call 512-478-5605

SEE US AT GDC 2003
Booth # 616
March 4-8, 2003 • San Jose, CA

www.renderware.com
Austin Derby Guildford Paris Tokyo

We're proud to announce RenderWare is now being used in hundreds of games and would like to congratulate our customers on generating over $ 2 BILLION of retail revenue from the titles released to date!

Game Construction Workspace



>>Powerful, Intuitive, Fast…

(i) Game View
(ii) Output Window
(iii) Target Link
(iv) Game Database
(v) Attribute Editor
(vi) Assets View
(vii) Behaviour Lister

Asset Management

Powered by alienbrain

1 The level designer downloads and executes the actual game to the target console using RenderWare Studio. A dynamic link is maintained for full interaction.

2 Source Code is parsed by RenderWare Studio to generate the UI that controls game parameters exposed by the programmer.

3 Objects can be drag 'n' dropped from the asset view into the game view. Behaviors can then be dropped onto these objects.

4 Object/behavior combinations are then instantly available for tuning gameplay in the attribute editor.

Game Code

Audio

- Dolby Digital & Pro Logic 1 & 2
- Outstanding multi-platform real-time audio
- Comprehensive sound bank management
- Virtual voice management

A.I.

- Powerful character & racing behaviors
- Multi-genre core capabilities
- Custom agent & brain scripting
- Powerful 3D pathfinding
- Time-slicing

Powered by kynogon

Physics

- Character physics
- Rigid body dynamics
- Flexible architecture & customizable pipeline
- Fully configurable joints (hinge, ball and socket, universal and skeletal)
- Vehicle optimizations (Q2)

RenderWare Core

PC

>>Get To The *Game*

Criterion
SOFTWARE

GAME PLAN

LETTER FROM THE EDITOR

M. M. Oh, Jeez

Online gaming has been one of the Internet's earliest and most persistent success stories, in both commercial and community terms — the evolution of massively multiplayer online games (MMOGs) from their MUD ancestors added a new dimension to the potential of the medium. For the second or third time in recent memory, 2003 was supposed to be The Year of the MMOG, with high-profile launches of The Sims Online (in December 2002) and Star Wars Galaxies (scheduled for later this year) bringing subscription-based gaming to the masses and — ideally — commensurate manna to developers and publishers. Arriving alongside these two giants is a slew of smaller, more diverse offerings that developers swear will see the light of day this year, and whispers of other big-name licensed titles to follow next year and beyond.

But with TSO faltering after launch, Galaxies scaling back on launch features, and both trying to retain their licenses' luster, the reality remains that online game consumers can afford to be fickle in the vast expanse of the online marketplace, to the detriment of developers who dutifully trawl message boards for feedback from fans on every conceivable nuance of a game before and after release. Certainly a strong emphasis on customer service is the right approach for developers to take for subscription-based business models, but I hope developers can maintain the line between providing good customer service on the subscription side, and avoiding the invariably dull result of trying to please all of the people all of the time on the design side.

It still takes a strong creative vision to engage and retain players in any medium, one that may not suit every potential mainstream user. I wonder if that is what has kept past successful MMOGs such as EverQuest and Ultima Online in niche fantasy genres. Ultimately players judge an online game's community as much as the game it surrounds; developers increase the variables by throwing the doors open wider. There remains a huge payoff for whoever gets the equation to balance out between a community's potential size and the members' sense of affinity that makes it a community in the first place.

Baca watch. Last year, many of you responded to my call ("So It's Come to This," Game Plan, July 2002) to write to your member of Congress and introduce yourselves and your profession. Not with any specific political agenda in mind, but as a means to open a line of communication between your representative and a part of his or her constituency that doesn't necessarily keep its hair-trigger finger on the moral panic button, that also contributes favorably to the local economy while nurturing a nascent art form right there in his or her district.

What prompted last year's column was Rep. Joe Baca's (D-Calif.) Protect Our Children from Video Game Sex and Violence Act of 2002, which was ill-timed on the heels of some other policy decisions unfavorable to the current state of game industry self-regulation. At the time it was introduced last May, it seemed likely Baca's bill would simply languish in committee and die, which it did in the House Judiciary Subcommittee on Crime, Terrorism, and Homeland Security.

At the time of this writing, Baca was poised to reintroduce a new version of last year's bill into the new Congress, which aims to criminalize retailers who sell mature-rated games to minors. One would be hard-pressed to find anyone who thinks that children should buy age-inappropriate games intended for adult sensibilities, but we're letting ourselves be lumped in with pornography (which is illegal under federal law to peddle to minors, for understandable reasons), not movies, television, music, and other mainstream forms of entertainment (which remain self-regulating). It's a harmful association that we need to combat.

Jennifer

Jennifer Olsen
Editor-In-Chief

GameDeveloper
www.gdmag.com
600 Harrison Street, San Francisco, CA 94107 t 415.947.6000 f 415.947.6090

Publisher
Jennifer Pahlka *jpahlka@cmp.com*

EDITORIAL

Editor-In-Chief
Jennifer Olsen *jolsen@cmp.com*

Managing Editor
Everard Strong *estrong@cmp.com*

Production Editor
Olga Zundel *ozundel@cmp.com*

Product Review Editor
Daniel Huebner *dan@gamasutra.com*

Art Director
Audrey Welch *awelch@cmp.com*

Editor-At-Large
Chris Hecker *checker@d6.com*

Contributing Editors
Jonathan Blow *jon@number-none.com*
Hayden Duvall *hayden@confounding-factor.com*
Noah Falstein *noah@theinspiracy.com*

Advisory Board
Hal Barwood LucasArts
Ellen Guon Beeman Monolith
Andy Gavin Naughty Dog
Joby Otero Luxoflux
Dave Pottinger Ensemble Studios
George Sanger Big Fat Inc.
Harvey Smith Ion Storm
Paul Steed Microsoft

ADVERTISING SALES

Director of Sales/Associate Publisher
Michele Sweeney e: *msweeney@cmp.com* t: 415.947.6217

Senior Account Manager, Eastern Region & Europe
Afton Thatcher e: *athatcher@cmp.com* t: 828.350.9392

Account Manager, Northern California & Southeast
Susan Kirby e: *skirby@cmp.com* t: 415.947.6226

Account Manager, Recruitment
Raelene Maiben e: *rmaiben@cmp.com* t: 415.947.6225

Account Manager, Western Region & Asia
Craig Perreault e: *cperreault@cmp.com* t: 415.947.6223

Account Representative
Aaron Murawski e: *amurawski@cmp.com* t: 415.947.6227

ADVERTISING PRODUCTION

Vice President, Manufacturing Bill Amstutz
Advertising Production Coordinator Kevin Chanel
Reprints Cindy Zauss t: 909.698.1780

GAMA NETWORK MARKETING

Director of Marketing Greg Kerwin
Senior MarCom Manager Jennifer McLean
Marketing Coordinator Scott Lyon

CIRCULATION

BPA INTERNATIONAL
Game Developer is BPA approved

Group Circulation Director Catherine Flynn
Circulation Manager Ron Escobar
Circulation Assistant Ian Hay
Newsstand Analyst Pam Santoro

SUBSCRIPTION SERVICES

For information, order questions, and address changes
t: 800.250.2429 or 847.647.5928 f: 847.647.5972
e: *gamedeveloper@halldata.com*

INTERNATIONAL LICENSING INFORMATION

Mario Salinas
t: 650.513.4234 f: 650.513.4482 e: *msalinas@cmp.com*

CMP MEDIA MANAGEMENT

President & CEO Gary Marshall
Executive Vice President & CFO John Day
Chief Operating Officer Steve Weitzner
Chief Information Officer Mike Mikos
President, Technology Solutions Group Robert Faletra
President, Healthcare Group Vicki Masseria
President, Electronics Group Jeff Patterson
President, Specialized Technologies Group Regina Starr Ridley
Senior Vice President, Global Sales & Marketing Bill Howard
Senior Vice President, HR & Communications Leah Landro
Vice President & General Counsel Sandra Grayson
Vice President, Creative Technologies Philip Chapnick

CMP United Business Media

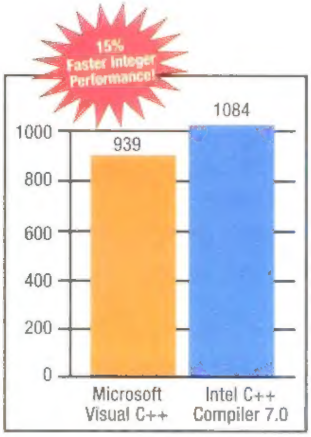
GamaNetwork

WWW.GAMANETWORK.COM

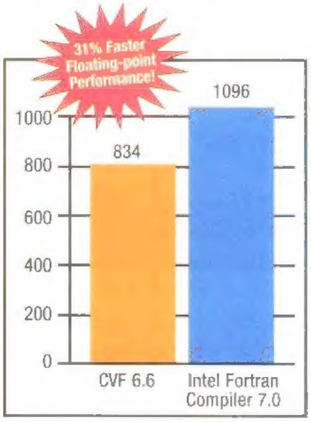
Maximize Your Application Performance

Intel® C++ and Fortran Compilers 7.0 for Windows* and Linux*

Intel's expertise in processors really shows in this latest release of their high-performance C++ and Fortran compilers. Intel compilers provide access to the latest advanced processor technologies. Give your application a performance boost without making source code modifications!



SPECint_base 2000[†]



SPECfp_base 2000[†]

Performance

Outstanding performance on Intel architecture including Intel® Pentium® 4, Intel® Xeon™ and Intel Itanium® 2 processors, as well as support for Hyper-Threading Technology.

Compatibility

C++

- Plugs into Microsoft Visual Studio* (6.0 and .NET*)
- Native source and binary compatibility with MS Visual C++*

Fortran

- Strong compatibility with Compaq Visual Fortran*

Support

Purchase includes 1 year of free product upgrades and Intel Premier Support

> *"The Intel Fortran Compiler 7.0 for Windows is a worthy successor to Compaq Visual Fortran and...remains head and shoulders above its competitors in both features and quality of implementation. This is a fast, efficient, and powerful compiler—a must-have tool for any Twenty-First Century Fortran migration or software development project."*
>
> *—Dr. Robert R. Trippi*
> *Visiting Scholar/Professor, Computational Finance*
> *University of California, San Diego*

[†] Intel® Pentium 4 Processor, 3.05 GHz, 512 KB L2 Cache, 256MB Memory, Windows XP Professional MP Kernel, Build 2600

YOU SAVE UP TO $117!	Paradise #	Retail	Discount
Intel® C++ Compiler 7.0 for Windows**	I23 0A10	$399.99	$305.99
Intel® Fortran Compiler 7.0 for Windows**	I23 0A11	$499.99	$381.99

** *Also available for Linux*

FREE 30-day trials available at:
programmersparadise.com/intel/gdm

Programmer's Paradise®
To order or request additional information call:
800-423-9990
Email: intel@programmers.com

03CT04C

SAYS YOU

A FORUM FOR YOUR POINT OF VIEW. GIVE US YOUR FEEDBACK...

Telling Stories

I wanted to share some ideas about Heather Kelley's "Narrative Games: Finding Another Side to the Story" (Soapbox, February 2003). I agree on some of the main points but would like to share an alternative viewpoint on some others.

While I agree that narrative is very powerful, I lean more toward the idea that games can be powerful by being generally expressive. For example, a painting can be expressive. It can directly evoke emotions with no apparent narrative. Consider Medal of Honor's first level, which is actually detached from the main narrative. While it tells a mini-narrative (soldiers storm the beach) the power lies not so much in the narrative (the story of soldiers storming the beach has been told many times) but in the raw emotional experience.

In the section called "Design for player expression," Kelley suggests that we create tools that allow players to "inject themselves into the story, supporting a broader freedom." Another approach to drawing the player into a game is taking control away. Again, look at the first level of Medal of Honor, considered by some to be a most memorable moment in gaming. Players have very little control over that level. They cannot shoot the German machine gunners, save their squadmates as they are mowed down, or even run too far left or right without stepping on a mine. During the first half of that level, players can only look around (the beginning tram ride of Half-Life is equally non-interactive.) While it may seem counterintuitive to take control from players, the point of a narrative or other expressive medium is to communicate ideas between people. Sometimes achieving this goal requires taking away some control from players.

Another way to make a narrative game easier to develop is to simply shorten games. A movie expresses its narrative successfully in two hours. A short computer game usually runs at least 10 hours. The idea that games should last at least 10 hours forces us to dilute the narrative with unnecessary action sequences, puzzles, or other beta-wave-type gameplay. Make games shorter and charge less for them. To make this economically reasonable, development houses will have to invest once in technology, then reuse the technology for many short, expressive games. Doing so will have the added benefit of helping with cash flow problems (less development time per title, more cash flow) and reduce risk for publishers (publishers can take a risk on a small, cheaper indie game if they are also publishing 10 other small titles at the same time).

While I agree that narrative is very powerful, I lean more toward the idea that games can be powerful by being generally expressive.

Also, we shouldn't make replayability a top-order goal. If our purpose is narrative, it's O.K. if the player only plays through once. I've only seen a lot of my favorite movies only once or twice. I don't need to play my favorite game 234,908 times.

Thank you for an informative article.

Alexander Jhin
via e-mail

Ratcheting up the Bar

I just read John Lally's article "Giving Life to Ratchet & Clank: Complex Character Animations" (February 2003). To make a long story short, using prototyping and MEL makes sense. After reading Lally's article, I thought, "Insomniac has just set the status quo for how other game developers are going to create." Why doesn't anybody else use logic when innovating the next level?

I really enjoyed the description of the use of use of prototyping, which as an artist sounds like a dream tool to work with. Thanks to Lally for disobeying Grandma and using translation and scaling commands.

Lally also mentions scripting facial presets using MEL. Like the article states, the intricate detail paid to facial expressions was a key factor in my interactivity with the characters.

Ian Johnston
via e-mail

Insomniac Inspiration

I just finished reading "Giving Life to Ratchet & Clank: Complex Character Animations," and I wanted to thank John Lally for writing one of the best articles I've read in the magazine.

I am currently setting up the character animation pipeline for a startup game development studio, and the article has given me many ideas on what I can do to streamline the process. Of particular interest to me are the processes for animating walk animations and setting up facial animation controls. It'll be interesting to see if I can simulate those tools for my project.

Danny Ngan
via e-mail

Let us know what you think: send us e-mail to editors@gdmag.com, or write to *Game Developer*, 600 Harrison St., San Francisco, CA 94107

brew™
2 0 0 3

April 28 - 30
San Diego, California

You recognize your future. It's a place where wireless carriers welcome your apps with open arms. Where millions of wireless users are waiting to snap them up. Sound inviting? Just attend the BREW™ 2003 Developers Conference, and learn how BREW helps you write apps and get them quickly to market. There are technical tracks, guest speakers, business sessions, and a wide open door to a bright future. Register at www.brew2003.com.

RUN TIME
BREW 2003
DEVELOPERS
CONFERENCE

Register Now:
www.brew2003.com

INDUSTRY WATCH

KEEPING AN EYE ON THE GAME BIZ | *everard strong*

3D file format forum launched. Intel recently launched the CAD-3D for Games forum on its site, www.intel.com, to aid discussions about an open, standard file format for 3D games. It will be hosted by Intel Developer Services and be monitored and moderated by the CAD-3D Working Group. According to the group, there are more 3D objects in the CAD world than anywhere else, and it hopes to set a standard file format that will result in a wider selection of objects for both game developers and architects.

PC graphics hardware sales increased in Q4 2002. According to reports from Jon Peddie Research, the PC graphics hardware market increased by 13 percent from Q3 to Q4 2002. JPR estimates that 53 million PC graphics devices were shipped in that time period, with the largest slice of the market going to Nvidia, who had 32 percent of the total unit shipments. Intel was second in units, and ATI was third. The growth of the hardware was spread evenly between mobile and desktop graphics chips and between graphics chips integrated with the motherboard and discrete add-in graphics system.

Infogrames unloads MacSoft. Infogrames recently sold MacSoft to Destineer, a privately held game developer and publisher owned by Peter Tamte, who started MacSoft in 1993.

Titles such as HARRY POTTER AND THE CHAMBER OF SECRETS helped Electronic Arts reach record revenues in Q3.

Electronic Arts posts record results, consolidates offices. Electronic Arts announced revenues for fiscal Q3, which ended December 31, hit a record $1.23 billion, up 48 percent over last year. Net income reached $250 million, up 89 percent from 2001. According to the company, this marks the first time a third-party publisher has reported a billion dollars in revenue in one quarter. EA said that 11 titles, including HARRY POTTER AND THE CHAMBER OF SECRETS, FIFA SOCCER 2003, MEDAL OF HONOR: FRONTLINE, LORD OF THE RINGS: THE TWO TOWERS, and seven others sold more than one million units during that quarter.

EA also announced it was consolidating offices, merging its Westwood Las Vegas, Los Angeles, and Irvine offices into one central Los Angeles location.

Games sales up, hardware revenue down. The NPD Group released year-end 2002 sales numbers for the U.S. game hardware, software, and accessories markets, showing that the game industry generated over $10 billion in sales, breaking 2001's record of $9.4 billion. Game software sales grew by 21 percent, selling 15 percent more units than in 2001. On the flipside, game hardware revenue declined slightly, from $3.7 billion in 2001 to $3.5 billion in 2002. Unit volume was up, though, by 10 percent.

Activision teams up with Dreamworks SKG. Activision has joined Dreamworks SKG in a multi-year, multi-property publishing agreement. The deal grants Activision exclusive interactive rights to a trio of upcoming computer-animated motion pictures: *Sharkslayer, Madagascar,* and *Over the Hedge,* with *Sharkslayer* scheduled for a 2004 release.

Send news items and product releases to news@gdmag.com.

THE TOOLBOX

DEVELOPMENT SOFTWARE, HARDWARE, AND OTHER STUFF

Havok unleashed. Havok recently released Havok 2, an updated physics tool for game developers. The update includes new character and vehicle kits, and, according to the company, the new version will offer up a tenfold speed increase on target platforms. Other features include toolchain integration amd cross-platform support.
www.havok.com

Turbo Squid to be publisher and distributor. Turbo Squid announced they will be the exclusive distributor and publisher for official Discreet 3DS Max plug-ins. Their first plug-in through the deal will be Cebas's finalToon, a cartoon and illustration tool that lets animators adjust line styles in realtime without having to re-render scenes.
www.turbosquid.com

UPCOMING EVENTS CALENDAR

GAME DEVELOPERS WORLD

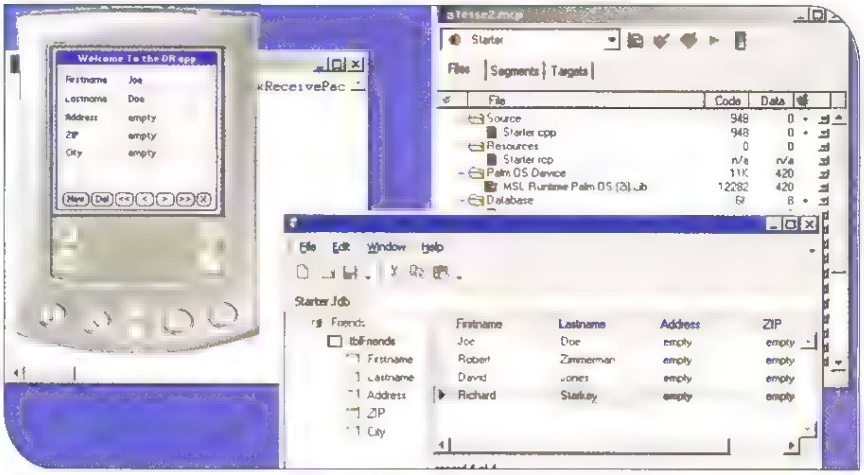
Bella Center
Copenhagen, Denmark
May 8–10, 2003
Cost: variable
www.gd-world.com

E3

Los Angeles Convention Center
Los Angeles, Calif.
May 13–16, 2003
Cost: Free–$550
www.e3expo.com

Metrowerks' CodeWarrior 9 for Palm OS

by justin lloyd



Welcome To the DR app

Firstname	Joe
Lastname	Doe
Address	empty
ZIP	empty
City	empty

File	Code	Data
Source	948	0
Starter.cpp	948	0
Resources	0	0
Starter.rcp	n/a	n/a
Palm OS Device	11K	420
MSL Runtime Palm OS (2i).lib	12282	420
Database	6?	8

Firstname	Lastname	Address	ZIP
Joe	Doe	empty	empty
Robert	Zimmerman	empty	empty
David	Jones	empty	empty
Richard	Starkey	empty	empty

CodeWarrior 9 for Palm OS is designed around a flexible plug-in architecture.

When considering an IDE, I look at every feature and every nuance; tools are a very personal agenda. Being a hardcore command line user, I enjoy using a properly configured makefile and build script, so what I want is an IDE that will perform the exact same actions, and have them be just as scriptable but presented in a more intuitive way than obscure text commands.

I've used Metrowerks' CodeWarrior off and on since the earliest Apple Mac days and on numerous projects for Game Boy Advance, Gamecube, PSX, and Playstation 2. For Playstation development it was always the tool of last resort, when you couldn't get your regular tools to talk to the DTL-2000.

Version 9 changes a lot of the underlying architecture from earlier versions, while leaving in place everything that is familiar to people who use it on a day-to-day basis.

CW9 is still showing its Mac roots, but Metrowerks is slowly engulfing them with its bigger goal: to create a unified host development environment for an undefined number of target platforms.

To achieve this goal, Metrowerks has adopted a flexible, and copiously documented, plug-in architecture. Their philosophy began appearing in earlier versions, and 9.0 completes the transition. Almost everything in the IDE, including the compiler, is a replaceable plug-in.

Seasoned developers want to see the IDE perform on a familiar code base, watch to see how it handles your project's unique issues. They realize that run-of-the-mill publisher samples don't really exercise the toolset. They have questions: What's it doing? How is it doing it? What are these options for?

There are a lot of Motorola 68K options, and very few for the new ARM. Also, the compiler only accepts inline 68K, even though it's possible to generate ARMlets, Palm's crippling attempt at exploiting the ARM.

Metrowerks has finally dropped Constructor by shipping the eminently usable PilRC resource editor. However, realizing that some developers are still using Constructor for projects under development, it is installed and available from the Windows Start Menu.

The IDE allows the import and export of configuration data. Manipulating IDE configurations via a script for a project is a feature that's extremely useful when needed. CW9 also stores the configuration data in easily parsed XML rather than an obscure, undocumented, ever-changing binary format.

The compiler generates code a little slowly. I didn't have time to perform proper timings other than cursory experimentation, but CodeWarrior takes longer to output the same amount of code than GCC for the same target CPU, with my GCC system performing more intermediate work. And still, GCC is several seconds faster. Looking over the assembly output, CW9 generates more CPU opcodes per source line, and uncapping the frame sync on my project didn't prove anything conclusive.

CW9 ships with two flavors of version control through plug-ins, VSS and CVS.

JUSTIN LLOYD | *Justin has over 18 years of commercial game programming experience on almost every released platform.*

Whichever solution you require, you still don't move completely away from the native toolset, the CW plug-in merely allows the IDE to talk to the back-end versioning system. CW9 can also integrate with Perforce and ClearCase via plug ins.

The usual way of developing for Palm OS is to run your code on an emulator/simulator, but the real world requires you to download to hardware regularly. Metrowerks has done its best by providing several ways to connect to the target. This is where CodeWarrior shows its strength. The integrated source level debugger supports C or C++, with native support for the Dragonball xZ 68K CPU line. There is no debugging support for either ARM assembly or ARMlets. Connection is made via the Hotsync cable utilising serial or USB. It's not possible to debug all hardware configurations, some devices do not allow remote debugging, the list of which can be found in the documentation. This incomplete coverage poses a problem as I'm currently developing an application for the Sony NX70V and am unable to debug either my GameCon peripheral or the camera code. CW9 supports an external debugger, such as Insight, running on top of GNU GDB.

CodeWarrior's editor has always been its biggest weakness, and version 9 doesn't improve on it much. The editor does provide most of the expected features, but half-heartedly. Code completion was clunky at best, unable to tell you either the parameters a function expects or the members a class or structure has defined; auto-indenting is primitive; syntax highlighting lacks sufficient granularity; brace and bracket balancing generates annoying delays in the default configuration and is fond of losing typed text; `open #include` works if you highlight the entire filename or just the base name, sans extension, as long as you don't have a file with "multiple extensions," for instance "foo.bar.h"; there's no "Delete Line" capability in the text editor.

The comprehensive Palm OS SDK documentation has been integrated in to the IDE. The documentation is accessible from the code editor, highlighting an API call or structure name and pressing F1 brings up the appropriate help page. The Metrowerks documentation covers a lot of ground, including embedded C and C++ usage.

CW9 includes the usual wizards for generating new projects, useful for creating a simple application or shared library. The documentation also includes comprehensive tutorials on using the wizards and setting up projects.

The IDE integrates a "graphical" file diff utility which is quite usable, generating the list of differences also in text.

One of CW9's most powerful features

is its support for Windows Scripting Host. Using any WSH 2.0 compatible scripting language, such as VBScript, you have automated control over the IDE and compiler. Coupled with the new command line features, this helps CW9 to integrate into a professional development environment.

Along with the standard Palm SDKs, Metrowerks ships Sony's, plus sundry others. It also includes the Palm Object Library (POL), an MFC for Palm without all the messiness. POL wraps the API calls and structures in logical C++ classes, but how wise this decision is I cannot say. I think the library needs more maturity and real-world usage before I would consider using it in a commercial project.

CODE WARRIOR 9 FOR PALM OS ★★★

○ STATS

METROWERKS
Austin, Tex
(800) 377-5416 or (512) 997-4700
www.metrowerks.com

PRICE
$399 for new users, $199 for upgrades

SYSTEM REQUIREMENTS
Intel Pentium or AMD K6 equivalent and 64MB of main memory, Windows 98/Me/2000/XP or NT 4.0 with SP6, CD-ROM drive for installation, and 380MB of free hard drive space.

○ PROS

1. Familiar environment when you move to other target platforms.
2. Large amounts of documentation including a well-documented plug-in architecture.
3. Source level debugger supporting C, C++, 68K, and ARM.

○ CONS

1. Incredibly primitive code editor.
2. More support for 68K than ARM.
3. Did I mention the incredibly primitive code editor?

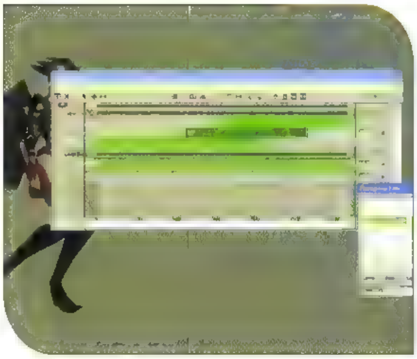
DISCREET'S CHARACTER STUDIO 4

by michael dean

Character Studio 4 is the latest release of Discreet's popular character animation package for 3DS Max. CS4 finally brings nonlinear animation (NLA) to 3DS Max. The interface to the NLA system is through the new Mixer; the Mixer panel contains a track-editing system similar to that found in the standard trackview, but the difference is that animators can control the motion of character bones (either all of them or a subset), through clips, layers, transitions, and weighting curves. While the interface is a bit unfriendly and not intuitive at first, it's very powerful, and can be creatively used to spawn a huge number of animations out of just a handful.

Another new feature is the integration of Biped data into the trackview. Movement, rotation, scaling, and controllers can all be applied to Biped objects and modified using their new standard curves in the trackview. Additionally, Discreet has added an animation Workbench to the interface. The Workbench is a version of the trackview, enhanced to provide features only available in Biped. These unique features include the new motion analysis tools, which can be used to scan over animations and check for potential problems such as motion spiking; fixes to found problems can be applied automatically, by hand, or completely ignored. As with the Mixer interface, all editing can be done while animation is being played, for immediate feedback.

The crowd system is another major new feature in CS4. Creating a crowd is a lot like creating particle systems, complete with influences, behaviors, and physics. The "particles," called Delegates, control the characters of your choice. The characters can be given a set of standard behaviors, and can also be given individuality and complex reactions to their environment via the Motion Flow network. Setting up this network is a lot like scripting: you give your characters several choices of which animation to play when dropped into an action/reaction environment.



The Motion Mixer is the heart of the NLA system. The red curve indicates the influence of one clip over another in a given track

A simple but welcome addition is a more contoured and less intrusive Biped skeleton. Animating while referencing the improved skeleton is much easier than with the old skeleton, as now you can tell at a glance in which direction everything is pointed. (The classic skeleton is still available to those who want it.)

The copy/paste functionality has been very much improved. Copying a pose or a posture now results in a clip being brought into a large clipboard (which can be saved to be used on any character), and displays each pose with a visual thumbnail. Copying a pose no longer overwrites the previously copied pose; now you can choose poses to paste from a simple list.

The Physique modifier has remained largely unchanged. Whether that is a good thing or not depends upon personal preference.

CS4 still is best in its class in terms of functionality, stability, and user-friendliness. The addition of NLA alone is well worth the price of admission ($995 plus a seat of 3DS Max 5.1 for Windows 2000/XP), and gives animators a lot more freedom and ability to create unique motion sets with a minimum of tedium and frustration.

★★★★½ | Character Studio 4
Discreet | www.discreet.com

Michael Dean is currently an artist at Ion Storm in Austin, Tex.

Gamania: 5 MMOGs in production

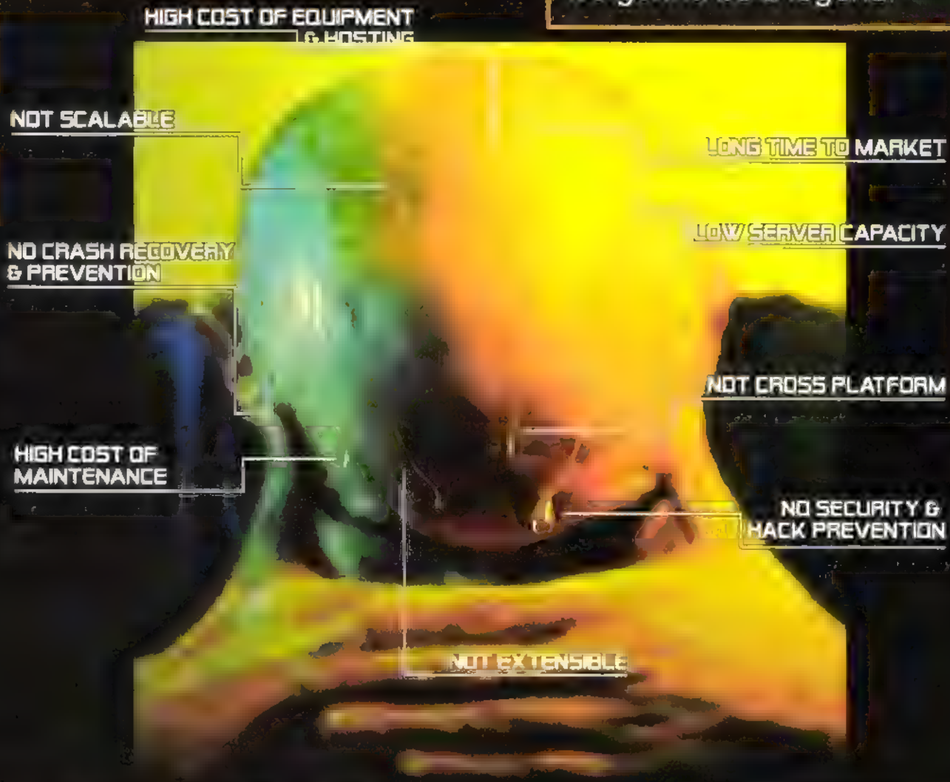
"Gamania is the most advanced online game operator in the world market today," said Albert Liu, CEO, Gamania Digital Entertainment, LTD. "With Zona, Inc's technology, we can upgrade the game player experience beyond the next level."

JoyOn: One secret RTS MMOG

"JoyOn Entertainment is Asia's leading general game publisher & developer," said Michael Cho, CEO, JoyOn Entertainment. "During our online game development, using Zona, Inc's technology, we have found the Zona network engine to be a highly sophisticated and unique technology which lives up to its name."

Capcom: Ghouls & Ghosts online

"I can't wait to play the MMOG version of 'Ghouls 'n Ghosts,'" said Yoshiki Okamoto, Chief Operating Officer Senior Managing Director Production Studios, Capcom Entertainment. "Bet it's gonna be a legend!"



MMOG failure is not an option. It can be, however, an unfortunate consequence of the unknown. And, more disheartening is the reality that those who will experience the game crashes and faulty game play are - your customers. Before you rush to implement a multi-million dollar MMOG network engine, consider alternative solutions that can help you scale, reliably, with less risk, for a lot less cash. When you've got the whole world in your hands, don't let your customers see you sweat.

THE FUTURE OF MMOG DEVELOPMENT IS IN YOUR HANDS_

For More infomation about **Zona, Inc.**, and our networking partners, please contact

Zona Headquarters
2606 Bayshore Parkway
Mountain View CA 94043
Tel 650-964-1133
Fax 650-961-8833
sales@zona.net

CE Technology Corp
12F-1 No 9 Sec 2
Luosfu Rd
Taipei 100 Taiwan
Tel +886 2 23961880
Fax +886 2 23961881

Zona Korea
Aname Tower 1219
702-10 Yeoksam
Kangnam Seoul
Tel 02-557-9360-1
Fax 02-557-9362

Zona Japan
AIOS Bldg 707
1-11-2 Hiroo, Shibuya,
Tokyo 150-0012 Japan
Tel 03-5798-2859
Fax 03-5798-2869

TERA ZONA

www.zona.net

PROFILES

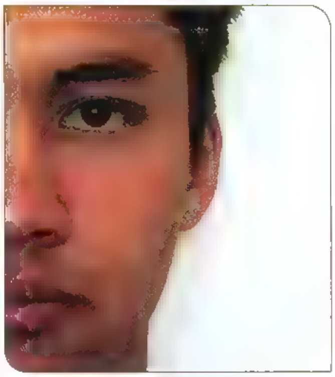
TALKING TO PEOPLE WHO MAKE A DIFFERENCE | *everard strong*

Be a Part of the Scene

Aaron Foo Talks about the Demoscene

If you're reading this while at the 2003 Game Developers Conference, you may have had the chance to check out the demoscene reel. It has proven so popular over the last couple of conferences that it's been made a prominent part of the newly installed Game Theatre this year.

Aaron Foo, currently a member of Sony Computer Entertainment America's R&D group, has also been an active participant and proselytizer of the demoscene movement, and was nice enough to give *Game Developer* answers to some questions we had about the scene, its members, and how it's affected the game industry.



Sony's Aaron Foo is a friend to the demoscene on U.S. shores

Game Developer: Describe, in brief, what the demoscene is all about, its history, and your involvement in it.

Aaron Foo: The short answer: It's a community of people fascinated by computers, digital art, real-time graphics and design, programming, 2D and 3D art, electronic music, and by people who enjoy fiddling with data bits on some obscure piece of hardware. The ultimate result of their tinkering is called a demo, a stylish, usually abstract, real-time visual treat synchronized to music. The demoscene has allowed many of its members to find and develop skills that often lead to a professional game development career.

The scene has been around forever, since people first started fooling around with the code on early gaming consoles.

I've been involved in the scene for about 10 years now (scary), and I still find time to participate in the Demoscene Outreach Group.

GD: The demoscene is very popular over in Europe where they host actual scener parties, but not in the United States. Why is this? Do you see this tide changing?

AF: Touchy subject! This has been the center of many flame wars over the years. Different people have different theories, but I honestly think it's a cultural and momentum thing. There are plenty of people with the dedication and skills to make demos here in the States, they just choose to apply those talents to something else.

The first U.S. demo party in many years will occur in August 2003. Hopefully this will kick-start more demoscene activity in the U.S.

GD: What can people expect from the demoscene reel at this year's Game Developers Conference?

AF: We'll have a good mix of the latest intros (size optimized), demos, wild demos (non real-time), and a few other completely crazy things thrown in. Ever wondered what 80×50 ASCII demos look like, or what you can do with 128 bytes? You're going to find out.

GD: What role does the GDC play in the demoscene's ongoing development?

AF: Events like the GDC and SIGGRAPH help show the raw talent of many people in the scene, and give them a foothold to start a career in the game development industry.

GD: There are different categories for projects to be classified into. Which ones are the most common?

AF: There's a typical 4KB intro category, but lately there's been a surge in 256-byte demos. I've also seen a few 128-byte demos around too.

GD: What were the early days of the demoscene like? Has the scene transformed along with the technology? If so, has it been for the good?

AF: The scene has definitely transformed from those days of people working on the Atari, VIC20, C64, the Amiga, PCs, and consoles. I think the early days were the best. Before the Internet we had a BBS to connect with each other, making it a lot more local. Now sceners create on anything that has a processor and a display device. Hopefully, the scene will never grow up, because half the fun of it is creating something for no other reason than the "cool" factor, to do it just because you can, and to impress your peers. If the scene grows up, and becomes serious and commercial, then it will most certainly die.

GD: What surprises you still about the scene and its members?

AF: How stubborn people are in sticking to prehistoric hardware and platforms; people are still writing demos for the Commodore 64.

GD: What are some of the lessons one can learn by being involved in the demoscene?

AF: The most important thing the demoscene teaches is, "Teach thyself." Much of the scene is about being self taught, learning and exploring how to do things on your own, without teachers, lectures, or reference books. This, I think, is the most valuable skill you can gain in the demoscene.

FOR MORE INFORMATION

The Demoscene Outreach Group: www.scene.org/dog
A showcase of 256-byte demos: www.256b.com
The Mind Candy DVD site: www.mindcandydvd.com/scene

Game Development. It's Serious Business.

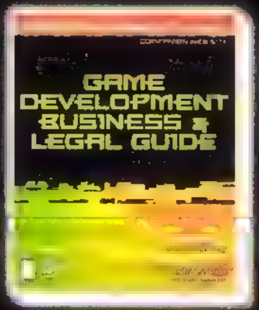
Check out a Premier Press GDC Sponsored Session:

The Trials and Tribulations of Running a Value Game Company
Thursday, March 7 3:00-4:00 PM

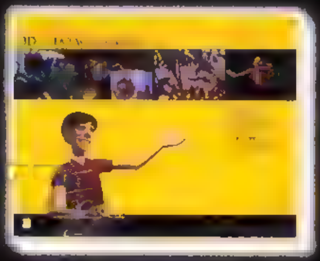
First Steps to Vertex and Pixel Shader Programming with DirectX
Friday, March 8 10:30-11:30 AM

Leveraging Online Resources to Make It as an Independent Game Developer
Friday, March 8 3:00-4:00 PM

Visit us at Booth 519 during GDC and pick up your free gift!



The Premier Press Game Development series.



Also from Premier Press, the Inspired series on 3D animation.

Premier Press
A Division of Course Technology
www.premierpressbooks.com
*Free gift available while supplies last.

Available at your favorite bookstore or call 1.800.842.3636 to order.

consumer reach

"[illegible] and an open application development platform. Of these phones, we expect roughly 10 million will be Series 60-based devices."

— Jorma Ollila, Chairman and CEO, Nokia
November 5th 2002

GDC and GDC Mobile 2003

- GDC Mobile Keynote presentation from Ilkka Raiskinen, Senior Vice President, Entertainment & Media Business Unit
- Get your hands on the latest devices and connect with Nokia's games team in the Nokia booth #808
- Check out Nokia's sessions during GDC and learn how to build games for the the N-Gage™ game deck and Nokia's platforms

SESSIONS: **Developing for The N-Gage™ Game Deck from Nokia**
When: Thursday, March 6, 10:15am - 11:15am and Friday, March 7, 10:30am - 11:30am
Where: San Jose Convention Center, Room [illegible]

forum.nokia.com/games

- Download tools and technical documentation on mobile gaming
- Get complete specifications and stay up to date on new devices
- Access global distribution channels for mobile games

www.forum.nokia.com/games

NOKIA

VOLUME 1 ISSUE 2 APRIL 2003
SPONSORED BY NOKIA

Game Developer Mobile

A bi-monthly supplement from the editors of *Game Developer* magazine

Hard Numbers Not at Hand

By Ben Calica

Phone companies are reluctant to release actual handset sales figures, giving developers no clue as to which model is the most popular

In an industry where every handset model represents development effort, the question of which ones will have the biggest market share is a basic one. In the culture of the wireless industry, however, developers are finding that getting answers to such questions is much more art than science

"The short answer is that we have no resource that gives us access to the handset sales numbers," laments Justin Siegel, president of game developer Jsmart "What we rely on is the buzz generated around the handset

That buzz comes from two sources, the handset manufacturers and the carriers. The handset manufacturers are in sales mode, motivated to create as much of a stir as possible about their latest models "The forward-looking [sales] numbers are a lot of guess," says Jan Lezny, senior director of developer/OEM relations for Qualcomm. "We sponsor external analysts The Yankee Group and Strategy Analytics to do reports on the market for our approved developers.

However, the group that really holds the key to sales predictions is the carriers, who decide not only which phone models to carry, but how much to subsidize each model of phone. "The number of handsets sold is directly related to what their cost is. Since it is subsidized by the carriers, they hold the final control," said Lezny

The way the carriers make deals with the handset manufacturers necessitates their keeping the sales numbers hidden. "They deliberately cloud the information from each other from a negotiation standpoint," comments Qualcomm's Lezny The deals they strike with the

BEN'S CALL observations from a mobile playground by Ben Calica

Nokia Rolls the Big Dice with N-Gage

When Nokia first announced the N-Gage back in November, I thought it was largely a ploy to get game companies to develop for Nokia's Series 60 platforms· If it did well, great, if not, Nokia still gets more games for their next-generation handsets. But at the lavish N-Gage launch, held this past February in London, Nokia made every effort to show that they're in the game for keeps. They still have the wireless industry culture of keeping information close to the vest, but here is what they revealed about their plans

1. **They think of this as a new platform.** For Nokia, this thing will work if it spurs on a new kind of gaming that blends in the ability to connect with a server or other players at any time. That's why they think they can take Nintendo; they are playing in a new type of gaming

INDUSTRY news

Ring Tone Sales Near $1 Billion Mark

According to Reuters, the sales of ring tones for wireless devices are approaching the $1 billion mark. A study released by London-based Informa Media Group said that the groups that collect royalties for the musicians had reported collecting $71 million in 2002, up 58 percent from the previous year. According to Simon Dyson, senior analyst at Informa, the figure, which usually represents between 10 to 15 percent of sales, indicates that ring tone sales total between $700 million and $1 billion annually. Both game and mobile industry executives have been looking at the success of consumers paying for ring tones as a bellwether for potential game sales

www.informa.com

5 Million Camera Phones in Japan

Japan's leading wireless carrier, NTT DoCoMo, has reported that it has over 5 million users for camera-mounted handsets. The fifth million

CONTINUED ON PAGE [illegible]

inside...

GameDeveloper
Mobile
VOLUME 1 ISSUE 2
APRIL 2003

EDITOR FOR MOBILE DEVELOPMENT
Ben Calica • bencalica@yahoo.com

A PUBLICATION OF GameDeveloper
EDITOR-IN-CHIEF
Jennifer Olsen • jolsen@cmp.com
MANAGING EDITOR
Everard Strong • estrong@cmp.com
PRODUCTION EDITOR
Olga Zundel • ozundel@cmp.com
PRODUCT AND INDUSTRY NEWS
news@gdmag.com

PUBLISHED BY THE GAMA NETWORK
A DIVISION OF CMP MEDIA LLC
PUBLISHER
Jennifer Pahlka • jpahlka@cmp.com
DIRECTOR OF SALES
Michele Sweeney • msweeney@cmp.com
415 9476217

CMP | GamaNetwork

> HARD NUMBERS CONTINUED FROM PAGE 1

handset manufacturers are based on potential unit volumes. If one handset manufacturer knows how many units of their competitors' handsets the carrier is going to buy, it lessens the carrier's leverage in the deal.

"We do work directly with the carriers, like AT&T and Verizon," says Oliver Miao, CEO of mobile game developer Centerscore. "Right now it really is based on which handsets are either currently out on the market or are coming out soon." Developers have learned their own ways of interpreting information from carriers. "Carriers will tell us, we expect X number of units to be sold by a certain date. But we typically discount those numbers because J2ME and BREW handsets have always been notoriously late to market," says Miao.

In addition, according to AT&T Wireless senior platform manager James Dierk, "The carriers will tend to buy small numbers of handsets initially to get a sense of how they will do in the market and then move more aggressively with the handsets that do well." Developers have found this situation hard to base predictions on.

Complicating matters is hitting the rolling target of technology advancement. Dierk estimates that "the black and while phones should be around for another 12 to 18 months before the colors replace them in the mass market."

One information source that is becoming increasingly valuable to developers is the new set of wireless game publishers. Because they see sales from a variety of titles over a variety of handsets, they have their own sense of what is hot. "Unfortunately, our predictions are based not on hard numbers, but instead on interest level of the publishers we work with," says Miao.

"The publisher is usually keen to target [the most popular handsets] first for commercial reasons," observes Colin Anderson, managing director of the U.K.-based developer Denki. Although he notes that sometimes publishers and manufacturers strike

case study

Sorrent: The Big Orr Dips Into Mobile Gaming

By Ben Calica

Scott Orr is a monster in the game business. Not a pop-out-of-the-closet-and-scare-you monster, but a bona fide hit-making machine to the tune of nearly $2 billion in sales over the course of his 20-year career. The best know of these was Electronic Arts' MADDEN series, which he headed up from 1991 to 2000. So why does a man who can choose to develop for any system that tickles his fancy leave the EA nest and start Sorrent, a company dedicated to mobile gaming?

"I started thinking about wireless during the latter half of 2000. I'd moved from MADDEN to EA.com, and we were trying to do a massively multiplayer title, and we started talking about wireless," remembers Orr. "Before that, we had created a fantasy football analyzer called EA SPORTS EDGE in 1998 and had looked into creating a wireless access for that."

When EA scaled back their online efforts, it gave Scott the excuse he was looking for to go out on his own. "We'd been spending a lot of time thinking about what server-based entertainment games would be like. When I started my company, one of the angel investors suggested looking at wireless," he recalls. What the investor spotted was the trend overseas for both improved handset capability as well as the beginnings of a market for content. "When I thought about it, a lot of the vision for what we could do would work well on this platform."

"Our first games were WAP-based," comments Orr. "WAP never worked for anybody anywhere, but the technology and application we built were good enough for us to get our first round of funding." It also let them build the beginnings of what distinguishes Sorrent from most of the other wireless game developers of this first generation. Their first game, SNAPSHOT LIVE FOOTBALL, was a throwback to the days of ZORK and text-based action display, but used the wireless phone and server technology to do multiplayer gaming. Most other game companies balked at the enormous latency problems inherent in mobile phone networks, opting for simple games with the online component being limited to the player downloading the game.

Sorrent's key licenses include the X-Files, FoxSports.com (shown), and a recent deal to release a Yao Ming–branded basketball game in China, where there are more than 200 million mobile phone users.

Barring more reliable data, developers are forced to rely on the buddy system for any kind of market size prediction in the mobile industry.

special deals designed to promote specific handsets.

The feeds that the publishers are looking at are particularly valuable, since they are based on real-time downloads. While they can't predict the future, they can gather a pretty good notion of the present. Barring more reliable data, developers are forced to rely on the buddy system for any kind of market size prediction in the mobile industry. Having friends among the handset manufacturers, the carriers, the publishers, and fellow developers is key to finding a sense where there is no concrete facts to be had.

To that end, CenterScore's Oliver Miao advises his fellow developers, 'Among the hottest handsets to develop for right now are the Nokia 7210, Motorola T720 J2ME, Nokia 3650, the Samsung A500, and the Motorola 95cl." From Jsmart's Justin Siegel's viewpoint, "J2ME current hot handsets in the U.S. are Nokia 6310 and 7310, and the Motorola T720. They have buzz."

> N-GAGE LAUNCH

The unit itself is pretty small and light, which is good for a combination portable console and phone.

2. They are trying to play in the traditional game business model. The N-Gage is not intended for downloading games like the rest of the mobile phone world. N-Gage games are distributed on 8MB-and-up multimedia cards that are produced by Nokia, with the appropriate fee taken out. You can charge for additional downloads, but it's a big step away from the download model that the rest of the industry employs. Still, Nokia is also pushing for the J2ME and BREW phones. (In fact they have committed to having J2ME on all their phones from now on.) It gets better. Other N-Gage features include a decent button layout for gaming and Bluetooth for local multiplayer gameplay without the standard wireless latency problems.

Nokia is also acting as a publisher, with both in-house and out-of-house titles considered, and money actually down on the table. In fact, in order to get to the really interesting information about N-Gage on their developer site, you have to pitch them and get into the next level of their developer program. For that you get to know

Sorrent, nevertheless, is jumping into the server-based gaming market with full force. "We're committed to doing multiplayer games via wireless technology, says Orr. But, he notes, "the speed we roll this out is driven by what the carriers are comfortable with. We've been working with them so they are comfortable with what they call network-aware applications. They want to make sure the initial rollout works on their systems. But we feel like he timing is right to introduce this next level to the gaming."

While Sorrent designs sports games and RPGs alike to suit the crucial five-to-10-minute session range, they are also taking advantage of the server-based nature of their titles to implement a new approach to dealing with both the games and the gamers. "We are creating a consistent digital persona [for the player]," says Orr. "When your skills are built over time in one game, you can take them over into another." That means that the physical ability that you built up playing football will come in handy when you need to wield an axe instead of a pigskin. "We think the real business is keeping a subscriber around. How do you get them to keep coming back month to month with what are essentially simple games? How do we give the player the ability to create an über-character?"

SACKED!

You've been sacked! Next down.

NEXT QUIT

It doesn't hurt that a number of the members of his 19-person team (14 of whom are engineers) are into EverQuest, or that a number of them were around EA when Ultima Online was being built. Orr is so sure Sorrent's cross-game approach is unique, they are even applying for a patent for it. Their server-based approach will ultimately enable players to download gamebots that will be tuned to the appropriate level of difficulty for the player, which Orr counts on as another way to extend the gameplay on what are essentially simple devices.

Orr discovered something unexpected when his company started doing serious development work on the mobile handsets: a great deal of fun. Recalling his mid-1980s work on the early consoles, Orr says, "It is like when you are sitting around with a bunch of friends saying, 'If only I knew then what I know now. Wouldn't it be cool?' This time we know what we are doing." He has found that the old-timers who know what is like to develop for very limited platforms are invaluable for doing this work. Very few members of his current have less than 12 years' experience.

It's not all going back in time, however. "We've been able to make use of many of the modern techniques," says Orr. "For example, we use the same kind of 3D tools we used on the Playstation, only it is rendered into 2D animation. In the old days we would have to use cel animation." Orr says such techniques give the company an edge graphically and help to cut down on the art path time.

Orr is optimistic about near-term growth for the mobile game market, but that doesn't stop him from designing for gamers. "We build for people who grew up with Game Boy, not some mythical mass market that may or may not buy games," Orr says. Sorrent tends to build on the higher-end handsets for that reason, rationalizing that the gamers like the better toys.

For brave developers now testing the wireless waters themselves, Orr advises, "Don't underestimate the challenge. Not only the challenge of how hard it is developing for these handsets, but don't underestimate the challenges of the network latency issues." As any of the old-timers will tell you, the simpler the box, the more clever the code. But, as Sorrent has learned, success also means identifying the design challenges and uncovering new opportunities of the medium.

> MIDP 2 0 CONTINUED FROM PAGE 1

beyond the listed specs, like the 104MHz ARM 9 processor, the 176×208, 4096-color screen (luxurious by phone standards), and the Series 60 platform with the Symbian OS. However, Nokia won't yet say what the cost of the games will be or what their cut will be

Nokia won't give any real installed-base projections, but they claim they won't do a new model unless it sells in the millions That is an interesting set of numbers for any developer. They currently have a major brand push to create the N-Gage. Even having left their marketing to the carriers in the last few years, Nokia is still the sixth most powerful brand, according to last year's *Businessweek*. They have signed up Sega, Eidos, Taito, Activision, and THQ as publishers to be there during their Q4 release this year. I asked them pointblank whether they had the patience to lose enough money to make this work, and up to the senior VP level, they said yes. Whether the company really does is the $500 million question

The first games shown at the launch event were good by cell phone standards, but in no way showed the unique or innovative aspects that will make the N-Gage a success as a gaming platform. And even though Nokia is getting better, the number of questions such as "What are the specific marketing plans and budget?" and "How much memory will be in the thing?" were met with artful, yet frustrating, dances. Whether Nokia clears the jump with tremendous success or crashes and misses in spectacular fashion, there is no question that they have their foot planted on the accelerator

next issue

- Inside the Minds of the Phone Barons
- The Revenge of the Old Timers
- Hot or Not; Insiders Pick Their Top Handsets

mobile developer resources

ATT Wireless Developer Resources
Main mMode Page
www attwireless com/mmode
Main Games Page
www.attwireless.com/mmode/features/games/index jhtml

BREW
Qualcomm site for developer forums, developer lab, BREW SDK, and Developer Directory
www.qualcomm.com/brew

Cingular Wireless Developer Program
http//alliance.cingularinteractive com

Forum Nokia
Nokia Developer Support. SDKs, J2ME developer information, Tradepoint marketplace for selling games to carriers
www forum nokia com

Motorola Motocoder Developers Site
www motorola com/motocoder

Siemens Developer Site
www.siemens-mobile com/developer

Sony Ericsson Developer Site
http //www ericsson.com/mobilityworld

Sprint PCS Wireless Developers Site
http://developer.sprintpcs.com

Symbian
Operating system for mobile phones
www symbian com

INDUSTRY news

CONTINUED FROM PAGE 1

came in just three weeks in January with sales of a new series of camera equipped handsets made by NEC, Matsushita, Sharp, and Fujitsu.
www.nttdocomo.com

Nokia Suite on Windows and Linux

The 1.1 version of the Nokia Developer suite for J2ME was released mid-January with versions for both Windows and Linux. This represents the first time the toolset has been released for the Linux environment. The suite can be used stand-alone, or integrated into IDEs from Borland and Sun. It's available free of charge from the Forum Nokia web site.
www.forum.nokia.com

iFone Sprints Classic Atari Titles

London-based iFone announced an agreement to bring "nostalgic" games from Infogrames' Atari library to Sprint Vision-enabled PCS phones. These games will include such classics as PONG, BREAKOUT, ASTEROIDS, and CENTIPEDE. iFone has locked up licensing arrangements for other Infogrames catalogues based on Infogrames-owned Hasbro, GT Interactive, and Ocean Software properties.
www.ifone.com

Immersion to Rock Your Phone

Immersion Corp., known for their touch feedback–enabled mice and other haptic technologies are turning their sights to mobile phones and PDAs. Their new cell phone prototype replaces the typical on/off pager motor with their hardware and control software. The vibration technology can then be used to provide physical feedback for games or patterned vibrations for custom ring tones. There are currently over 100 haptically enabled devices and more than 500 games on the PC market.
www.immersion.com

MEF Tries to Smooth Trails for MMS

The Mobile Entertainment Forum is launching an initiative to develop policy guidelines for the new Multimedia Messaging Services (MMS). "Our experience from the premium-rate industry has shown that a clear regulatory framework is crucial for the development of new markets and services," said MEF member Patrick Naughton. The wireless industry is hoping to both repeat and extend the wild success of text-based SMS messaging on the last generation of mobile phones.
http://www mobileentertainmentforum.org

Send news information to news@gdmag.com.

Unified Rendering LOD Part 2

Last month I started building a unified LOD system. My intention is to create a generalized method of LOD management that works for environments, objects, and characters. Traditional LOD systems tend to be complicated and can impose difficult constraints on mesh topology or representation. To prevent such impediments, I want to make the LOD system as simple as possible.

I chose static mesh switching as the underlying LOD method. Last month I discussed the basic technique of generating blocks at various levels of detail and building seams between the blocks to prevent holes from appearing in the world. But this technique alone is insufficient; switching between static meshes will cause visible popping in the rendered image, so I need to address that problem.

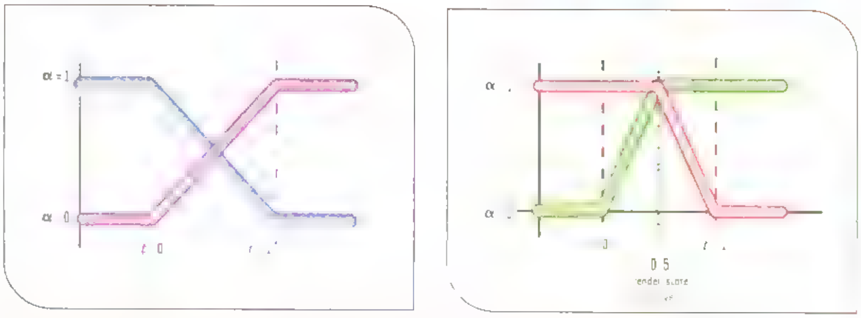


FIGURE 1A (left) The usual cross-fading alpha blend function; it is not suitable for use here, since it doesn't result in the rendering of opaque terrain. FIGURE 1B (right) Blend function modified to ensure that at least one block is opaque at all times.

Preventing Popping

There are three methods that are often used to prevent popping. I'll call the first method "incremental transitioning." The idea behind incremental transitioning is to pop only small subsets of a mesh each frame, in the hope that the small pops will be nearly invisible. Continuous LOD and progressive mesh systems employ this idea. However, as I treat all meshes as atomic objects in my algorithm for maximum simplicity, incremental transitioning is not possible here.

The second method is geomorphing, in which we move the vertices slowly from their positions in the low-resolution shape to their positions in the high-resolution shape (or vice versa). The third method is color blending, whereby we draw the block at both levels of detail and interpolate between the resulting colors.

Deciding between geomorphing and color blending, I chose color blending. I'll justify my decision later, but first I want to talk about the basic implementation of the color blending technique. My later justification is necessary because color blending may at first glance seem wacky and inefficient.

Color Blending: The Basic Idea

With color blending, we want to transition between LODs by rendering each LOD individually, then blending the output colors at each pixel. On DirectX 8–class hardware and earlier systems, we would do this using alpha blending to cross-fade between the meshes, while doing some tricks to ensure that reasonable values end up in the Z-buffer.

Given DirectX 9 or above, with multiple render targets, color blending becomes easy. So I'll concentrate on the trickier implementation with DirectX 8 and earlier.

The basic method I use for the blending was recently reintroduced to me by Markus Giegl (see For More Information), though I swear I saw it a while back in some publication like the *ACM Transactions on Graphics*. We could imagine naively cross-fading between the two LODs; this would involve drawing one LOD with an alpha value of t, and the other with alpha $1 - t$ (Figure 1a). Neither mesh would be completely opaque, so we'd be able to see through the object to the background. That's not a workable solution.

Giegl proposes altering the cross-fading function so that one of the meshes is always opaque (Figure 1b). We fade one mesh in, and only once it becomes completely solid do we begin to fade the other mesh out.

I do things differently from the way Giegl proposes in his paper. When drawing the translucent mesh for any particular block, I found that if I left the Z-buffer writes turned off, unpleasant rendering artifacts occurred, since distant portions of the translucent mesh often overwrote nearby portions. We could solve this problem by sorting the triangles in the translucent mesh

JONATHAN BLOW | *You can contact Jonathan at jon@number-none.com. Rock over London; rock on, Chicago. Bandini is the word for fertilizer.*

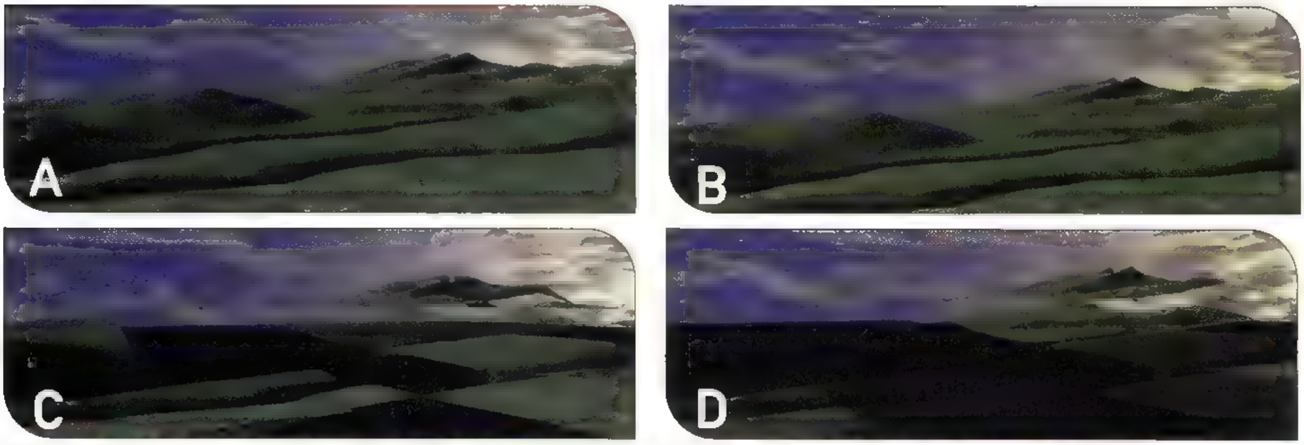


FIGURE 2A (top left) A simple terrain scene. FIGURE 2B (to) The non-transitioning blocks of the scene are rendered once, opaque, with a simple shader The "fading-out blocks (represented by the red line from Figure 1) The fading-in" blocks (represented by the green line from Figure 1) Note that 2b and 2c make a complementary set of blocks that together consist of the whole terrain, so the total amount of duplicate rendering can be quantified by looking at Figure 2d

by distance, but that's a very slow process. Instead, I render the translucent mesh with Z-buffer writes enabled. Technically this is still not correct, since self-occluded portions of the translucent mesh may or may not be drawn. But on the whole, this incorrectness is unobtrusive. Giegl's paper suggests disabling Z-writes for the translucent meshes, which I cannot believe produces good results for nontrivial scenes.

It's important that I render the translucent mesh after the opaque mesh; otherwise the Z-fill from the transparent mesh would prevent portions of the opaque mesh from being rendered, creating big holes. This rendering order creates an interesting problem. When the blend function in Figure 1b switches which mesh is opaque, I need to change the order in which the meshes are drawn. At first A is transparent and B is opaque, so I draw B first, then A. Then A becomes opaque, so I draw A first, then B. Interestingly, no consistent depth test function can be used to prevent popping. Consider the pixels of A and B that have the same Z values; that is, the quantized intersection of A and B.

If we render the meshes with Z-accept set to <=, then these intersection pixels will be filled by A immediately before the switch, and filled by B immediately after the switch, causing a pop. If the Z-accept is <, then the pixels where Z is equal will show as B before the switch, and A afterward. To circumvent this problem, I switch the Z function when I switch the mesh rendering order. Before the switch-over, I render with Z-accept on <=, after the switch-over, I render with Z-accept on <. Thus the intersection pixels are always filled by A.

We will still have some Z-fighting after we have completed all these steps, because we are rendering a lot of intersecting geometry. But in general the Z-fighting doesn't look too bad, since the LODs tend to be similar. On higher-end hardware, we can increase the precision of the Z-buffer to mitigate this problem.

A terrain scene like Figure 2a will contain some blocks that are transitioning between LODs, and some that are not. First, I render nontransitioning blocks as completely solid; these are very fast, since we're just doing vanilla static-mesh rendering (Figure 2b). Other blocks are either "fading in" (Figure 2c) or "fading out" (Figure 2d); each of these types of blocks is rendered translucently after the corresponding opaque mesh is drawn.

If we're not careful about rendering order, we will have problems where we render a translucent block and then a solid block behind it, causing pixels in the solid block to Z-fail. To prevent this problem, we can render all the solid blocks first, and then render the translucent blocks back-to-front.

You might think that color blending would be much slower than geomorphing, since we are rendering more triangles for transitioning objects, and rendering twice as many pixels. But as I'm about to show, the vertex and pixel shaders for color blending are simpler and faster. As it turns out, the cost for geomorphing can approach the cost of rendering geometry twice.

Geomorphing: The Basic Idea

The most straightforward way to perform geomorphing is to interpolate the vertex positions every frame on the main CPU, then send the resulting triangles to the graphics hardware. This method results in slow rendering; to render quickly, we want all the geometry to reside on the GPU.

With modern vertex shaders such as DirectX 9's, we can interpolate the geometry directly on the hardware. To do this we must store position data for both LODs in the data for each vertex, because vertex shaders provide no way of associating separate vertices. Then we use a global shader parameter to interpolate between the positions.

This vertex shader will be longer and slower than a shader that renders a non-geomorphed mesh. Hopefully, much of the time we will be drawing non-geomorphed meshes, and we only activate geomorphing during the short transition from one LOD to another. So we will write two vertex shaders, a slow one and a fast one.

That doesn't sound so bad yet, but suppose we want to render animated characters instead of static meshes. We need a third vertex shader that performs skinning and such. But now, we also need a fourth vertex shader that performs geomorphing on meshes that are skinned.

In the end, we'll end up writing twice as many vertex shaders as we would in the absence of LOD. And don't forget that we need to maintain those shaders and handle their interactions with the rest of the system throughout the development cycle. That's not nice. Combinatorial explosion in vertex and pixel shaders is already a big problem, and geomorphing seems to exacerbate it.

The capability for branching and subroutines is being introduced into vertex shaders, and this may help deal with the combinatorial explosions. But it's too early to say for sure how speed in real games will be affected, and thus whether the resulting shaders will be useful overall.

Next I'll look at the problems that can occur when these LOD methods interact with other parts of the rendering system.

Texture Mapping and Shader LOD

As geometry recedes into the distance, we will eventually want to use lower-resolution textures for it. If the mesh is made of several materials, we'll also want to condense those into a single material; otherwise, we will render only a small number of triangles between each rendering state change, and that's bad.

In general, at some level of detail we will want to change the mesh's texture maps and shaders. If we do this abruptly, we'll see obvious popping.

Geomorphing doesn't help us here at all. If we want to transition smoothly between textures, we need to build some blending logic on top of geomorphing, making the system more complicated. Since we perform pixel-color logic twice and blend, our pixel shaders will slow down, perhaps to a speed comparable to the color blending method. That makes sense, because we're performing a big piece of the color blending method in addition to geomorphing.

The color blending method by itself, on the other hand, handles texture and shader LOD automatically. We can use different textures and texture coordinates and shaders for any of the levels of detail; the LOD system just doesn't care. It's completely unconstrained.

Normal Mapping

Suppose we are using normal mapping to approximate a high-resolution mesh with lower-resolution meshes. Ideally, we would like to decrease the resolution of our normal maps proportionally with distance from the camera, just as with texture maps. But even if we give up that optimization, there's another problem that makes geomorphing unfriendly to normal mapping.

When performing lighting computations, we transform the normal maps by tangent frames defined at the vertices of the mesh. When geomorphing, we need to smoothly interpolate these tangent frames along with the vertex coordinates. Tangent frames exist in a curved space, so interpolating them at high quality is more expensive than the linear interpolations we use for position. If the quality of the interpolation is too low, the results are ugly. So our vertex shader becomes more expensive — perhaps more expensive than the color blending method, which renders 1.25 times the number of triangles that geomorphing does, but with simpler shaders. (This figure of 1.25 is representative of a heightfield-based scene; it will change in future articles.)

In stark contrast to the combination with geomorphing, normal mapping and color blending get along very well together. The differing LODs can be covered with different normal maps and parameterized arbitrarily. In fact, we could elect to eliminate normal maps on the lower LOD entirely.

Stencil Shadows

One nice thing about geomorphing is that it's possible to implement stencil-buffer shadows without undue difficulty. Because the geometry changes smoothly, shadow planes extruded from the geometry change smoothly as well. That's an advantage over color blending.

Suppose we want to use stencil shadows with color blending LOD. The simplest approach is to choose one of the rendered LODs of each block to generate shadow volumes. But when the level of detail for a block transitions, its shadows will change discontinuously. To avoid this, we would like to represent fractional values in the stencil buffer that we could somehow use to interpolate the shadows. Unfortunately, the stencil buffer algorithm doesn't work that way.

For stencil shadows to work with color blending requires DirectX 9-class hardware or above. We would use two different render targets to generate two sets of stencil values, one for each level of detail. Then, at each pixel of the visible scene geometry, we compute a light attenuation factor by interpolating the results from these two stencil buffers. This technique is nice because it is highly orthogonal to our mesh representations and shaders. On a DirectX 8 card, using this LOD technique would produce stencil shadows that pop. But stencil shadows in general are most viable on next-generation hardware, anyway.

Sample Code

In this month's sample code (available for download from the *Game Developer* web site at [illegible]), you can move around a simple terrain that has been cut into blocks. The color blending method of LOD interpolation has been implemented to prevent popping.

FOR MORE INFORMATION

Giegl, Markus, and Michael Wimmer. "Unpopping: Solving the Image-Space Blend Problem."

[illegible]

Looking Outside the Game

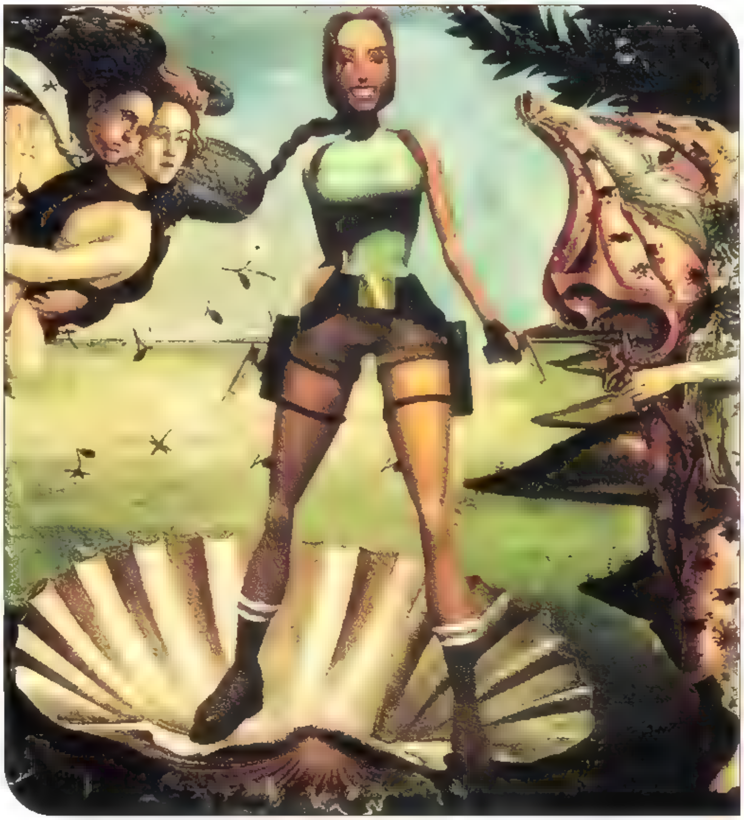


Illustration by Steve Munday

The video game industry is only really a few decades old. Ignoring its vague origins in the 1960s, videogames haven't been around for much more than 20 years in a meaningful, mainstream kind of way. Looking around at developers (and here I mean those that make the games, not the suits in big offices), most of the workforce are in their 20s; a growing number are now in their 30s, but very few have seen their 40th birthday or beyond.

The craft of game development itself as well as those who work within it indicate an industry very much in its infancy. It's impossible to visualize the changes that will occur 100 years from now, but I can fully imagine that today's games will be viewed with both the respect and the amusement with which we now watch the earliest silent movies.

As pioneers at the forefront of an emerging medium, we are in a privileged position. We have relatively little baggage, and our industry is constantly focused on the future and how to deliver the best gaming experience possible with our current level of technology. As this technology is always moving forward, the horizon will always remain somewhere off in the distance, and it is difficult to see a time when this will not be the case.

Pause for a minute, however, and remember that technology is only one aspect of game development; a game is the product of the expertise of many people who combine their skills across a number of disciplines. Technology may dictate much about how a game looks, sounds, and feels, but it has no input as to actual content. It may tell me I can't put more than six characters on the screen at once, but it won't design or animate them for me.

Technology may give me the tools to create the game world, but it will be of little help when it comes to deciding how that game world should look. For this kind of help, we can of course look at other games, but perhaps more sensibly, we can choose to look outside the game industry to see what we can learn form the world that exists independently of our screens.

Just about every artist I have ever worked with has been able to cite a number of people that have influenced them. Looking at the work from which others draw inspiration tells you a great deal about how a person sees the world. I'd like to list some areas I believe can be of direct benefit to a game artist, all of which have contributed to my understanding of art, both in and out of games.

HAYDEN DUVALL *Hayden started work in 1987, creating airbrushed artwork for the games industry. Over the next eight years, Hayden continued as a freelance artist and lectured in psychology at Perth College in Scotland. Hayden now lives in Bristol, England, with his wife, Leah, and their four children, where he is lead artist at Confounding Factor.*

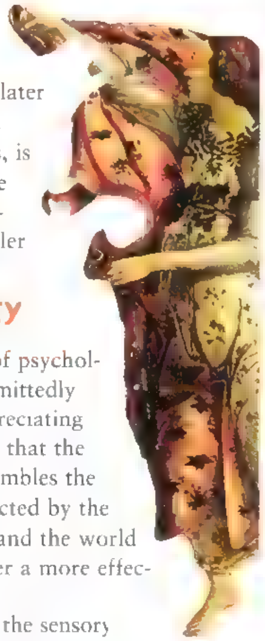
Photography

Photography is about getting a message across visually, and while the message can be as simple as the beauty of a sunset, it can also be as emotionally charged as images of the Holocaust. With a successful photograph, all of the component elements combine in a single static image to have the artist's desired impact on the viewer.

A photographer works within a certain frame, as does a game artist. While a game is generally not static, and most often takes place within a 3D space, there is still the opportunity to create a visual impact from setting a scene, balancing light and dark, foreground and background, and focusing the player's eye on what is important (or what you want them to think is important). Just as you view most photographs from eye level, gameplay takes place from the player's point of view, particularly in the case of a first-person shooter. This perspective can help maximize the visual quality of certain areas that players encounter as they move around the environment.

Sculpture

I think there is a case to be made for the relevance of sculpture to the game artist. Consider the process of box modeling or subdivision surface modeling, where a shape is essentially fashioned from a primitive volume. Admittedly, in sculpture the removal of material leads to the end result; adding volume isn't really what it's about. In principle, though, the idea of creating form from a simple initial piece of material (real or virtual) by refining shapes first broadly, dealing with mass and proportion, then later adding detail and other refinements, is equally applicable to both the sculptor and the modeler.



Psychology

Many areas of psychology are admittedly dubious, but appreciating some of the ways that the human mind assembles the information collected by the senses to understand the world can help us deliver a more effective game.

Today, most of the sensory input from a videogame is visual, followed by audio, with limited controller vibration adding a small amount

of tactile feedback. A game artist naturally has direct access to what players see and therefore controls the quality of their experience to a large extent. As much research as has gone into the human visual system and how our mind decides what our eyes are looking at, we can gain much wisdom by researching the subject of perception.

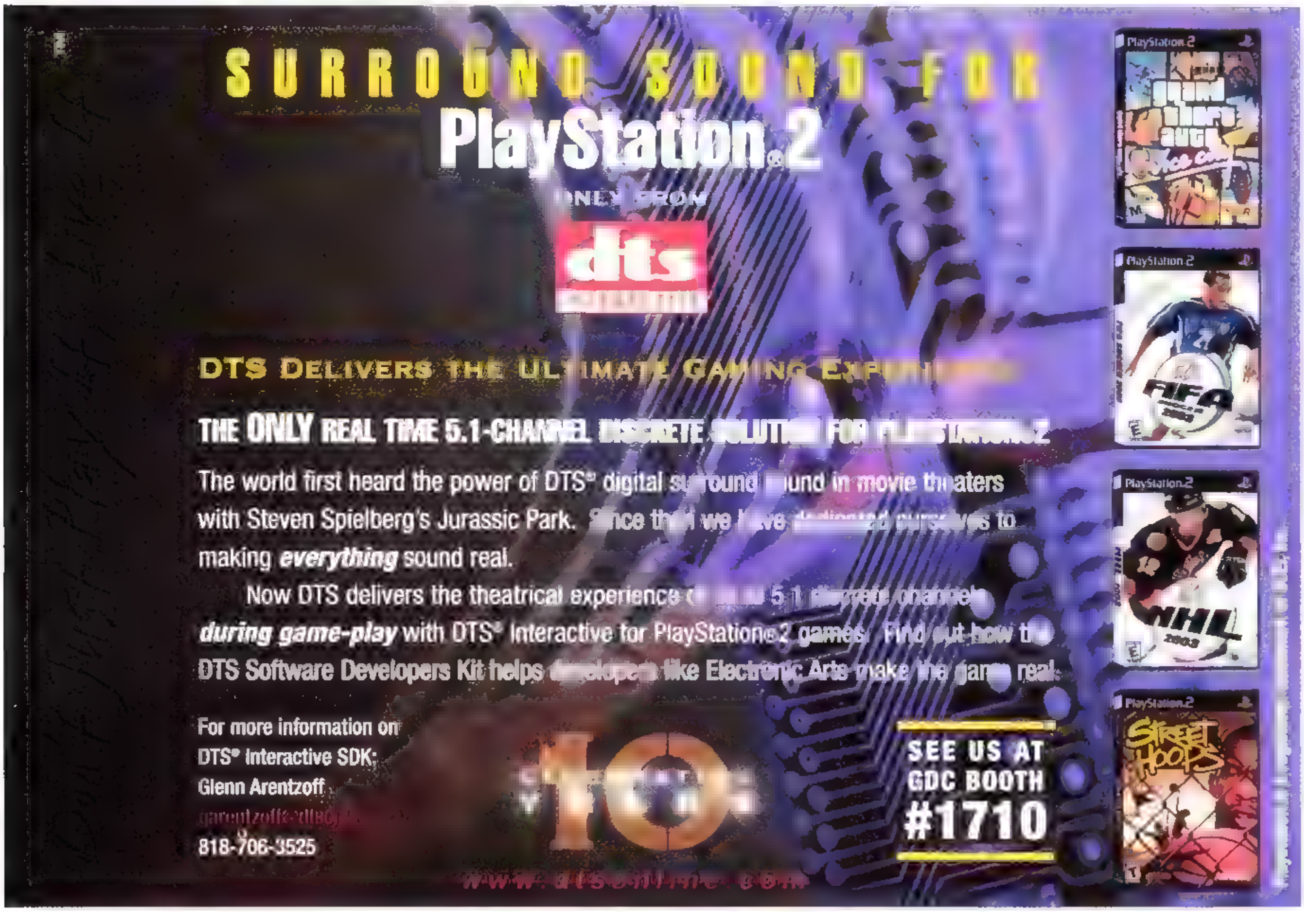
Memory also falls under the banner of psychology. Certain game types require the player to remember vital information, or where certain locations are in relation to each other. Studies into the way we store and retrieve information can help us with design decisions that enhance rather than hinder a player's experience.

Psychology also encompasses areas such as body language, attraction, and facial expression. Researching these fields of study can add an extra dimension in the areas of character design and animation, helping to convey emotion more effectively and making a character appealing (or unappealing) to the player.

Industrial Design

One of the joys and sometimes difficulties of being a game artist is that from one project to the next (unless you get stuck with never-ending sequels) you can be called upon to create a vast spectrum of different worlds and character types.

Sometimes your game might be pure fantasy, giving you as much creative freedom as you could wish for, but it is fair to say that often you will be working on a game that's set in the future. Regardless of whether this future setting is recognizable, or whether it's one that is far removed from what we know, you'll likely need to create a wide range of vehicles, weapons, and hardware. An understanding of industrial design can help you with this process. We are lucky, in that functionality within game design only has to be the appearance of functionality, as no one is ever going to have to make or use a real version of anything we put on screen. As a result, we can be a little more inventive. However, taking our imagination too far without any regard for how things may need to work if they were real often produces something that's not convincing to the player.



One area of interest under the umbrella of industrial design is that of ergonomics. Ergonomics derives its meaning from the Greek word *ergon*, which translates as work, and *nomoi*, which means natural laws, and is the study of how human characteristics can be integrated with the design of devices or systems. In our modern world, this kind of design is all around us. With advancements in fabrication techniques, miniaturization, and new materials, we tend to see a move away from objects whose design is dictated by their function and toward objects that are designed with the comfort and aesthetic tastes of the user in mind.

If we are building a future world for our game and hope to make it feel believable to players, an increased emphasis on ergonomic design is one factor that can help with this goal. Looking at the design evolution of anything from handguns to aircraft can provide worthwhile illustration of this kind of thinking, as it shows a clear progression of ideas and presents a solid foundation from which to extrapolate future design possibilities.

Art History

Obviously the work of contemporary artists, particularly those involved in fantasy and science-fiction illustration, can be a direct source of inspiration to those working on games that cover similar territory. But is there any value to be had in looking farther afield at art that may seem unconnected to games?

On the one hand, some of the extremities in modern art are a little too abstract to be specifically relevant. The paint splatters of Pollock, Picasso's cubism, and the wavy poplar trees of Van Gogh have plenty to recommend them (not least their multi-million dollar price tags), but I don't quite see them having much of an impact in the game industry.

However, take some time to consider the landscapes of Constable, the sunsets of Turner, and Rembrandt's dramatic use of chiaroscuro: the forms and colors that are beginning to emerge in the next generation of games, those which have more room to maneuver visually, are beginning to reflect some of these great works. The richness of these paintings can now serve as a more direct inspiration to game artists. The way in which artists have captured beauty and drama on canvas over the centuries may be radically different from that of a modern-day game artist, but the emotions we are attempting to elicit are the same. Only the context has changed.

Rembrandt's use of chiaroscuro in such paintings as *The Prodigal Son* can serve as inspiration for game artists looking to expand their horizons.

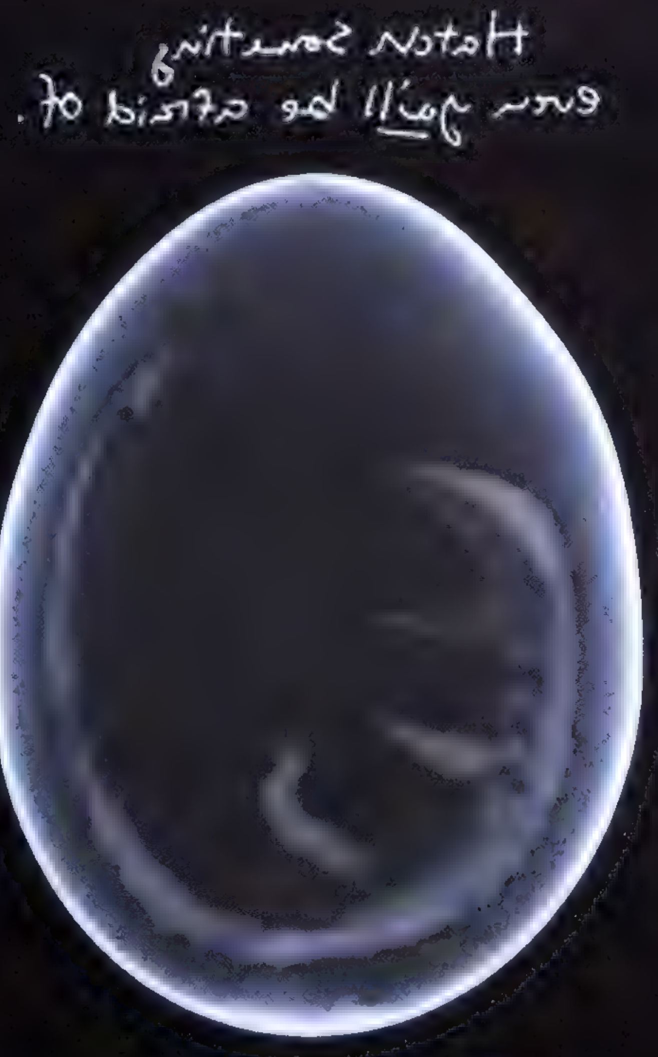
For a more direct link to games, we can look at the Art Nouveau movement from the late 19th and early 20th centuries. Stylistically, Art Nouveau has been used as a point of departure in much of the design work for science-fiction films and illustration for many years. Take away the Art Nouveau influences from *Star Wars* or *Star Trek*, for example, and you will be left with a great deal of empty space.

The stylized organic forms of Art Nouveau, coupled with geometric patterning that emerged through the work of designers such as Charles Rennie Mackintosh, have been adapted many times over the years to symbolize a future design ethic. It contains both order and chaos, as well as a slight retro feel with which we are now familiar.

Tromp l'oeil, a style of painting that attempts to trick the observer into thinking that the flat surface of the image is in fact three-dimensional in some way, has been around for a very long time. Traced back as far as the fifth century B.C., tromp l'oeil grew in popularity through the 17th and 18th centuries, and found a significant resurgence in the last century through the photorealist movement. It is true that bump and displacement mapping, together with the increase in actual geometry at game artists' disposal, have reduced the appearance of fake depth through texture work. However, it remains necessary (at least at present) to effectively present the player with the illusion of depth on what are generally flat surfaces. Examining how tromp l'oeil art uses light, shadow, and perspective to produce the appearance of depth is still valuable, as most texture artists will need to use similar techniques to make in-game surfaces more interesting.

Keep Looking

We as game artists must remember the many areas that have something to offer us in the way of help and inspiration. Games don't exist in isolation. Looking beyond the games that have been released over the last year or so can present creators with ideas that are fresh and more interesting in a market that is already crowded with 20 cloned versions of every good idea.


Hatch Something
even you'll be afraid of.

"The Interface": Making Peace with Your Producer

When I got into game audio I had no idea what a producer was, and no one ever sat me down to explain their role. Not only do producers have to maintain a grip on the dynamic game development process (and the more forward-looking your design, the more dynamic the process), but they also have to be as intimately familiar as possible with everyone's job.

Not knowing this, a lot of audio folks assume that producers and their ilk are the spawn of hell. They slash features without discussion, they ignore quality as a matter of course, and they ruthlessly put down audio as taking a backseat to graphics. Sometimes, this is true. I won't deny that, and I pity the poor souls who have to work in such circumstances. Most of the time, however, what looks like demonic evil is really misconception and ignorance on both sides. Producers aren't experts at audio, and audio folks aren't in the middle of the scheduling and budget trenches. With this in mind, I want to share learned several lessons aimed at enhancing the audio professional/producer relationship.

Lesson #1: Take initiative. Assume that producers and project leads aren't going to walk into your studio or call you up and politely say, "Make me as familiar as possible with what you're thinking the audio should do for this game and how you plan to do it. I have loads of time." Take the initiative to educate yourself about the project, its needs, and your team's wants, and then bring forth the result to the higher-ups. It's not going to happen by itself.

Lesson #2: Do your homework. Let's begin with your ideal situation on a project: You're burning with dreams of 192kHz, 32 bit interactive music, the latest compression schemes, Lexicon-quality dynamic reverb, and brilliant multichannel 3D positioning, across all platforms. You have visions of the most intuitive and feature-filled of toolsets to achieve these goals, summoned at a whim by the programmers. The awards recognizing your genius pop on to your shelf from near and far. Back on planet Earth, you actually need to think carefully about how to get to that audio Fantasy Island, and few of us, including myself, do this well.

Ion Storm audio engineer Mark Lampert presents his ideas to Bill Money, producer of DEUS EX 2

Once again, communication and research are key. Talk to your programming leads. Talk to the producers about goals, milestones, and deadlines. Talk to the design leads about what will fit with the game's vision. Be realistic about what will really help your title — 6.1 for a puzzle game? Subtle fading of ambient tracks during a rip-roaring F-1 racing game? Come on. The more intelligent your decisions are, the easier it will be to convince the producer and other leads that you can make the audio for the game like a well-oiled machine, with little maintenance. Once you've carved your plan and presented it efficiently to the producers, these gatekeepers of your holy audio grail are far more likely to lower the drawbridge to future ventures, even riskier ones.

Let's take another example: If you're pitching dynamic music to a producer, you can't just say, "It'll sound better," and expect to get a positive response. You need to be aware of exactly what the music will do, and more importantly how it will make the title you're working on shine. Will there be layered tracks? Will those layered tracks really help your sports title? Try creating a few examples in your sequencer. Even though they may not be real-time, you can show how the music will interact outside of your brain. If you find you need either more memory or a bigger streaming buffer to handle the additional data being controlled, in addition to the logic that will govern what the music does, you'll need to outline this behavior for programmers and producers alike. Will there be crossfading? This will also involve handling of additional data, all of which takes time (however little it may seem) to code, and resources in the game-data pie chart, which is split up between you, programming, and art at the highest level.

Lesson #3: Don't be afraid to ask questions. Too often people (once again this includes me) are afraid that asking too many questions will get them a swift boot up the backside. This is hogwash. Sure, if you're moments away from a deadline and there's an issue that isn't resolved, asking too many questions as opposed to taking action will probably result in frustration. But especially in preproduction, find out everything you need to know based on your goal, and then make your decision.

Keep all these lessons in mind the next time you strike out upon the long road of your next project, and the result can be both better game audio and happier developers all around.

ALEXANDER BRANDON | *Alex is the audio director on DEUS EX 2 at Ion Storm Austin and is gathering old game soundtracks for a massive compilation. He is also the membership director for the Game Audio Network Guild (www.audiogang.org) and is on the advisory board for DirectSound 9.*

Beyond "Save the World"

This month's rule harks back to the very first "Better by Design" column (March 2002), which introduced the rule "Provide Clear Short Term Goals." I mentioned that the rule was trumped by "Provide an Enticing Long-Term Goal," but did not explain in detail what that meant — until now.

The Rule: Provide an Enticing Long-Term Goal

Many (but not all) games benefit by having an ultimate goal that is made clear to the player fairly early on. Making this goal enticing is one way to pull the player into the game world and encourage passion.

The Rule's domain. This rule applies most strongly to story- and character-based games, such as RPGs and action adventure games. Old standbys of this type of goal involve "Save the world from destruction," or "Kill the evil wizard," and, of course, the classic "Save the world from destruction by killing the evil wizard." MMORPGs provide dozens of long-term goals to keep players subscribing for years. The rule is important to strategy games and vehicle simulators too, where the long-term goal may be to build a civilization, win a war, or gain critical promotions while pursing a personal career. The rule is not quite as important for sports games or storyless FPS games, where the implied long-term goal of winning the match, race, or tournament takes over. Finally, it applies only weakly to puzzle games like TETRIS or BEJEWELLED, where the long-term goal can be as basic as "Get the high score." THE SIMS thrives without an explicit long-term goal, but its very familiar real-world setting and gameplay invites players to provide their own long-term goals.

Rules that it trumps. The rule does not actually trump but rather augments "Protect the Player's Suspension of Disbelief," as the long-term goal is a crucial way to draw players into a fictional experience – which is precisely why the domain of this rule is strongest with styles of games that involve story and characters.

The opening of MAX PAYNE engages players with the story early on

Rules that it is trumped by. "Provide Clear Short-Term Goals." An enticing long-term goal is not as important to the player's immersion and enjoyment of a game as a clear short-term goal. The elegant way to blend these two rules is to start the player off with clear short-term goals and let the long-term goal be explained slowly on the side. Even better, let the long-term goal grow organically out of the progression of short-term goals

This rule is also trumped by "Provide Story Reversals," a rule from narrative fiction. The majority of novels and films contain a reversal (often several) where the protagonist's initial goals change midstream. Reversals propel the story along by sending it off in a new direction before the audience (or player) has a chance to become bored with a predictable plot.

Examples and counterexamples. MAX PAYNE begins with a very cinematic opening, introducing us to the title character and inviting us into his head as he finds his family killed by criminals. This event provides a strong pull for vengeance, as in the "Now it's personal!" parlance of Hollywood. In fact an allied rule may well be to "Make it personal," to provide that extra motivation to the player. It's a cliché, but only because it has repeatedly proven to be effective.

Another example would be the story line in the single-player campaign of STARCRAFT, which provides clear short-term goals (attack this Zerg before it destroys your siege tank) within mid term goals (complete this scenario by establishing a base), while slipping long-term goals and reversals into the gameplay (get revenge on the general who betrayed you, save humanity from the Zerg menace).

Puzzle games, as I mentioned, do not fall far into this rule's domain, as they are typically about as far from narrative gaming as you can get, but it's intriguing to look into gaming's distant past and see the media hype and lip service that the rudimentary cutscenes of MS. PAC MAN provided, or more intriguingly to see how the old classic Macintosh game THE FOOL'S ERRAND managed to unify a group of disparate puzzles with an artfully told story line.

Be careful to remember the word "enticing" in this rule, and realize that what entices one player may bore another. One reason reversals can be so intriguing in a story line is that the people who were not very motivated by the first goal may become swept up in the second.

Remember, the ultimate long-term goal for game designers is to entrance and entertain the player. More on that in an upcoming column.

NOAH FALSTEIN | *Noah is a 23-year veteran of the game industry. His web site, , has a description of The 400 Project, the basis for these columns. Also at that site is a list of the game design rules collected so far, and tips on how to use them. You can e-mail Noah at noah@theinspiracy.com.*

copperhead™

Introducing Copperhead™ Technology, a powerful, flexible suite of game development tools and technologies for PlayStation® 2, GameCube™, Xbox™, and PC

FOCUS ON FUN

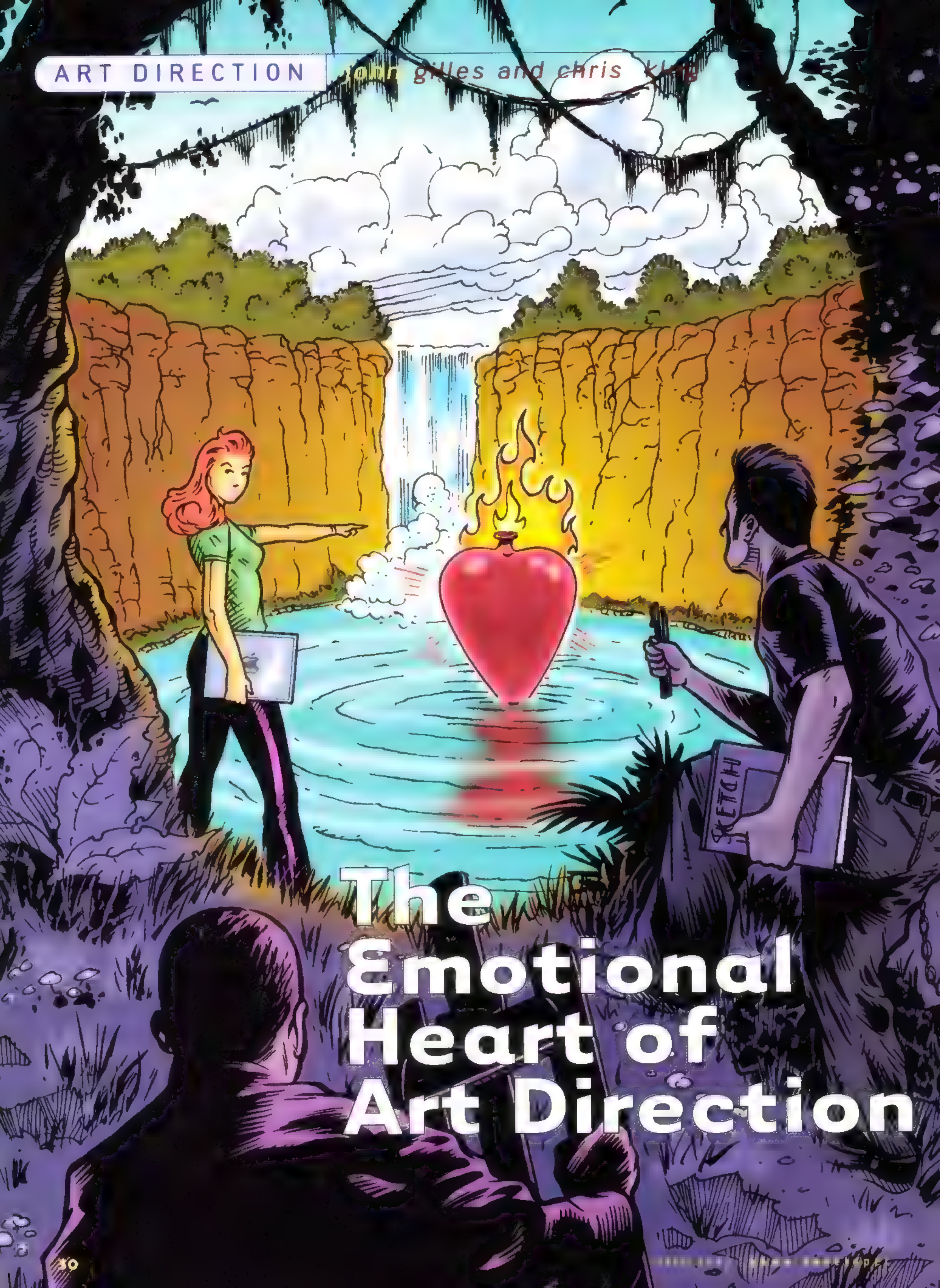
Touchdown Entertainment

Check out Copperhead and Jupiter at the Game Developer's Conference
ExpoSuite 134

www.touchdownentertainment.com

copperhead™ jupiter

john gilles and chris king



The Emotional Heart of Art Direction

How Visual Design Processes in Traditional Media Can Be Used to Improve Visual Design in Games

As games begin to deliver the visual quality of movies and television and also approximate the visceral experience of live theater, the demands made on the game industry's art directors grow. Fortunately, the techniques employed by art directors in other media can teach art directors in games a trick or two that will elevate the visual and artistic quality in our industry.

Designers in traditional media mostly approach their assignments, regardless of what they might be, pretty much in the same manner. They begin by searching for the "emotional heart" of the piece they are designing. Once they determine what that is, they base their design decisions upon it. They may be designing sets, or lights, or costumes; they may be working for the Rolling Stones or for Dame Judith Anderson; they may be working in the Metropolitan Opera House on the Upper West Side or at Soho rep on Mott Street in lower Manhattan; the audience may be convicts in a maximum security prison or children in a park in San Diego. These details influence many technical considerations and often affect the manner in which the design expresses itself, but they rarely alter the designer's search for the emotional heart of the piece.

Are Games the Same?

When Chris worked in the pencil-and-paper game business, regardless of what kind of game he was designing (RPG, war game, parlor game, strategy game), the graphic artists he worked with proceeded with their job pretty much as he had done while working as a professional lighting designer. Whereas they might have used the word "tone" to express what we refer to in this article as the emotional heart, they indeed searched for it, and their graphic designs expressed the tone for the game they all worked on.

Emotional centers of traditional dramatic pieces are easy to find; most scripts are created with a theme in mind, because writers tend to work that way. Examples of possible emotional hearts of well-known works are *Star Wars* (belief in yourself can overcome all obstacles); *The Matrix* (you are what you believe yourself to be); *Blade Runner* (all life is sacred, whether born of woman or manufactured); *Unforgiven* (rational violence leads to irrational violence); and *The Lord of the Rings* (even the smallest person can make a difference). However, since dramatic work is subject to interpretation (which is why one production of *Hamlet* can look so different from another), these examples may not match exactly what an author thought the theme was when he or she wrote the screenplay. Rather, these interpretations are our guesses at the theme from watching each piece.

However, after Chris began working in the computer game business, and especially after graphics cards became capable of displaying 3D graphics and higher-resolution textures, he observed that some of the artists he worked with did not talk about the emotional heart of the game, nor did they seem interested in finding it.

These artists were most certainly concerned with the design of the game. They were all working toward making sure the game looked its best. They all had great pride in their work, all were very talented, and all approached their designs professionally. But the only yardstick they used to measure their success or failure seemed to be how "cool" the graphics looked. Whether their visuals were in sync with the tone of the game design, whether they supported the overall feel of the game design, and whether they had anything to do with the emotional content of the game design were sometimes lost in the process. What was even more disturbing was the fact these talented artists never even discussed these issues.

These artists and designers were all talented, and their designs were often brilliant. It wasn't that they wanted to avoid these issues, as Chris discovered, but they did not have the training to think in a way that revolved around the emotional heart of a work, nor did they have the vocabulary to talk about such issues.

Chris couldn't figure this out. How had these artists been trained? What was their thought process? How did they organize their design? How did they find a style or visual language for any game they worked on? How did they know what to include and, more importantly, what to leave behind? How did they determine what was "cool"?

The more Chris asked around, the more he discovered a common thread. These artists and designers had been trained in various schools and worked with many great teachers. Some were self taught. But none of them came from

JOHN GILLES AND CHRIS KLUG | *John and Chris are lucky enough to have worked in both games and traditional media. Chris has been a professional game designer for 22 years, and before that, a professional lighting designer in the arenas of theater, opera, and rock and roll. John has been an art director and designer in theater, television, and cinema for 23 years. The authors met in graduate school while studying theatrical design at Carnegie Mellon University in Pittsburgh*

DEUS EX is an example of a game that delivers experiences closely mapped to those found in traditional dramatic media.

training in the dramatic arts of cinema, TV, and theater. Both authors of this article have been trained as designers in theater, and it was clear to us that the more games moved towards 3D environments with high-resolution graphics and increasingly humanlike avatars, the more game settings became environments in which virtual actors played out their stories against virtual sets. Games were clearly beginning to deliver experiences (MAX PAYNE, METAL GEAR SOLID, FINAL FANTASY, MEDAL OF HONOR, DEUS EX) that more closely mapped to those found in the more traditional dramatic media. We began to explore how the training we had received in college and the traditions developed in those arts could be applied directly to game art direction.

We began by examining what differences and similarities existed among the media. Was there anything in the very essence of game art direction that argued for throwing out the old methodologies? How were the development processes different, and how were they the same? Could game art direction learn anything from the traditional media, or were they such different beasts that learning one from the other was a waste of time?

Art Cost vs. Capability

Since the cost of games is increasing seemingly without limit, art directors have a responsibility to find development methods that control those costs. We do not have the easy excuse that movies have, namely that the star actors are charging exorbitant fees. Our costs are rising mainly because the quality of the visuals is getting better as hardware becomes capable of displaying more photorealistic images rendered in real time. The graphics card makers aren't going to slow their advances in chip development, and so the market will demand that visuals get better and better in step with the newest hardware. Simply put, the game industry needs to get better at creating environments, both to reduce development time and minimize manpower needs.

How do we do this? One way is to understand the techniques that art directors have known for years in movies and TV (where sets can only be built once), and adapt and apply them to designing our 3D environments. Avatars are nothing more than actors in front of a set, after all. Thus the lessons learned from centuries of theater and decades of movies and television set and costume design can surely apply to 3D universes and the avatars that inhabit those universes.

How Traditional Media Are Generally Built

Did you ever wonder as you're watching the latest blockbuster movie, "How did they come up with that idea?" or, "Why did they do it that way?" Unfortunately, just as often we may ask, "What idiot thought that one up?" The origin of all of these comments leads back to the movie's script. In TV, movies, and theater, everything starts and ends with the script. And in TV, movies, and theater, the art directors turn to the script to find the work's emotional heart.

It is even more true in television and theater than with movies. In television, writers rule the world. They often get billed as producers or executive producers, but those are just designations for the main creative force on a show, who is almost invariably the writer. Aaron Sorkin gets billed as executive producer on *The West Wing* and also gets writing credit for the episodes he writes. And as this past season's failure *Girls Club* proved, David E. Kelly can get any show green-lit simply by the power of his involvement. Both solitary authors. Singular visions.

In theater, the playwright is everything. In fact, it is standard language in the Dramatists Guild agreement that not a single line of a play may be changed without the playwright's written consent.

All this explanation isn't meant to glorify writers; it's to illustrate a major difference between the way settings are designed in games as opposed to traditional media: a single author versus authorship by committee. It is the design-by-committee characteristic of game development that hamstrings game art directors from streamlining their processes and focusing their vision.

In order to help game art directors overcome this challenge, we must ask ourselves, Why does the game business operate that way? What other game development methodologies contribute to this paradigm? And how does this make the art director's job impossible?

The Pyramid of Development

Cinema, television, theater, and game development all share a development pyramid, created and honed over time as the businesses matured. In each case, the pyramid's height is the time it takes from inception to delivery of product to audience. The pyramid's width is the amount of dollars being spent at any corresponding time of development.

A movie's pyramid (Figure 1) is narrow at the top and very broad at the bottom. Since most of the production cost of a movie happens while it is being shot and in post, movie studios have learned that they should take their time developing the script. They came to know that the stronger the script and the more thought given to the script details, the more cost-effective the actual shooting time would

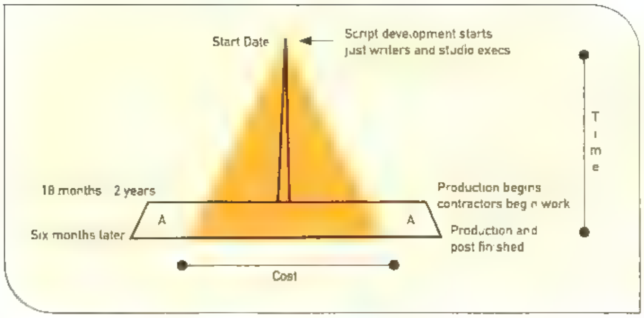


FIGURE 1 A film's cost pyramid

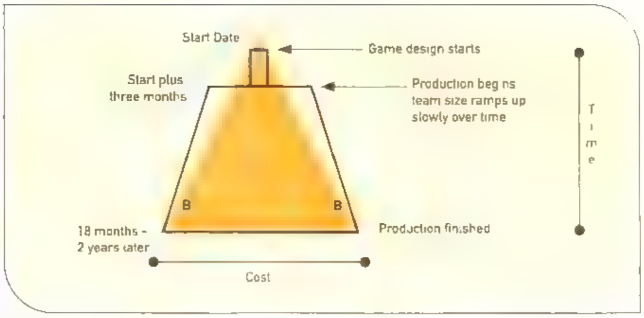


FIGURE 2 A game's cost pyramid.

be. So, the development of the script can take years, but the money spent during those years is minimal, because it is the work of a few individuals and mainly of one person, the screenwriter. Only when everyone has agreed on the script (which is no simple task) does the real money get spent (stars hired, sets built, and so on). Cinema has a slow, cheap ramp-up and then spends 80 percent of its money in a very short period of time toward the base of the pyramid.

Game development's pyramid is very different (Figure 2). While it is almost always true that more money is spent at the base of the pyramid than at the top, the difference in width is not as great as with film. This is because most games update their engine technology for each new title, and there is an enormous amount of pressure to get the whole team working as soon as possible. Often game development studios have artists and programmers sitting around with nothing to do until the game design is finished, and then they can rush off and start banging away. It's important to note that almost all of these game developers waiting to start are full-time employees, while most of their equivalents in the movie business are contractors. In that business, the cost of labor doesn't begin until the script is ready. In games, the cost does ramp up, but it is always burning at a relatively steady rate.

Hence the design of the average major game title is done in a hurry, so as to get the development team working as soon as possible. Owing to this over-arching priority, game designs do not get the same time to gestate fully before construction must start, which, in turn leads often to half-baked ideas from which the art director must then start designing.

More importantly, because time isn't taken at the beginning of the process, parts of development teams can easily get out-of-sync with each other regarding what game they're doing. Since time constraints ensure that the design document often isn't anything more than a bare-bones outline when programming and art direction starts, it is entirely possible, and altogether common, that the designer will realize in mid-stream that some parts of this thing just don't work with each other.

These and other well-known production bottlenecks tend us toward our art directors throwing darts at a target called "cool," because there is nothing else of substance at which they can aim at the time.

Taking Artistic Control

Given what we know about the way things are, it's up to game art directors to develop techniques that will help them develop and execute a clear artistic vision within the limitations of game development's design-by-committee chaos.

First, and most important, as art director insist you attend the earliest design meetings. If you have to, agree to keep quiet during them. While it is much better if you can actively contribute to these early discussions, the reality in many studios is that design staff can be a little territorial. In those situations, beg, wheedle, or cajole yourself into those meetings. Most experienced producers or executive producers will want you there, as they will want you working on sketches and concept art as soon as possible. To those designers who are a little threatened, I suggest that you convince them that you're not there to suggest or interrupt, just to listen. If nothing works, well, they're going to have to tell you about the game at some point, so be patient and wait for them to spill the beans.

Once in these meetings, listen. Not so much to what the game system is, but to what feeling the design staff might be trying to get at. Try to get at the core of the game in as simplistic a way as possible. Is the game dark? Is it light? Is it masculine? Feminine (unlikely with all the testosterone pooled in the halls of game development houses)? Fast? Slow? Gritty? Polished? Hot? Cold? You get the idea. If you can talk with a designer who can think visually, try to get him or her to talk about what he or she wants the game to feel like. Often you'll have to search for what is really behind the designer's words, because he or she might be the kind of designer who can only think numerically or systemically. Be persistent. Don't let the discussion drill down too deep; no specifics now, just feelings. Good designers will be able to communicate this way, and bad ones won't. If you can't get much in the way of volunteered information, try asking questions that force the designer to make choices, like, "Does the game feel dark or light to you? Dirty or clean? Smooth or rough?" Most designers will be able to answer these questions.

The odds are pretty good that while the systems and environments may come and go during the game's development, this core feeling the design team is going for won't change much. In fact, there is a secret technique to keep yourself more on track with the visual design than most game designers will ever manage because they won't use this technique. The game's designers may stray and fumble and go in six different directions, but you'll be working away right at the core of the game.

This technique could be the most important tool you'll have as an art director on any project. It will save your soul, guide your hand, focus your efforts, and even do your dishes. Chris used it with every design he ever did in professional theater, and John uses it as well. Every working production designer we know uses this technique in some form. The technique is simply to pick a single visual image to be your concept for the piece. It has to be a visual image of some kind: a painting, a still from a movie, a photograph, a sculpture, a physical item like a piece of kitchen ware, a lamp, a pair of pants, something. The thing should have all the qualities you need to guide your design: line, form, color, and feeling. The item you choose should evoke a feeling within you.

How It Looks, How It Feels

Using the movie examples and their emotional hearts given previously, we could come up with the following visual images: For *Star Wars* (belief in yourself can overcome all obstacles), the visual image could be a single white lily growing out of a cracked obsidian vase. For *The Matrix* (you are what you believe yourself to be), a silvered mirror encased in the cheapest black plastic frame. Reflected in the mirror is Neo, dressed in vibrant, bold colors. Standing in front of the mirror is the actual Neo, dressed in grays and muted earth tones. For *Blade Runner* (all life is sacred) the visual image might be the a unicorn, silvery mane flowing, strong and powerful, running through a garbage heap. For *Unforgiven* (rational violence leads to irrational violence), it could be a square wooden frame, artfully carved, symmetrical in all ways. The frame sits inside an iron carpenter's vise, which presses on the frame at two of its corners. This pressure causes fractures and faults in the frame's joints and members. Splinters and fragments of the frame drop off and fall on the bleached desert sand. And for *The Lord of the Rings* (even the smallest person can make a difference), the visual image could be a giant blood-red stone sphere, ancient and worn, standing on top of a tiny forest-green pebble. An infant, clothed in a soft gray diaper but nothing else, sits balanced at the apex of the sphere. No matter which way the infant leans, the sphere will tilt to that side and fall off the pebble and roll in that direction.

The issue isn't whether these images we just made up for these movies are the best images in every case. What we're illustrating is that from those images, the art director could begin to make choices about line, color, size, weight, and other design elements for everything from the avatar's clothing to the buildings in the virtual world.

Once you have chosen this visual metaphor, you use it to guide your design so that everything going into the game follows, supports, evokes, mirrors, or complements this item you've chosen. If some idea comes along that a member of your staff is pitching, or an exec has one of those brainstorms, run the idea through the sifter of this image. If it doesn't match the style or feeling of this image, it either has to get thrown out or changed so that it does.

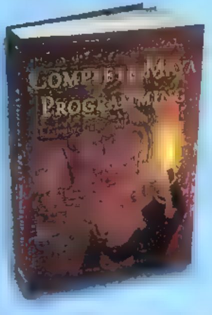
One interpretation of the emotional heart of Clint Eastwood's *Unforgiven* was that rational violence can lead to irrational violence

To show this might work, suppose that in our *Star Wars* example, the producer really wants to see the bad guys dressed in red. But the image you've chosen seems to indicate they would be dressed simplistically, black and shiny with sharp edges. Methods to reach a compromise could include black shiny armor with red piping on the sharp bits, or red shiny armor with black undergarments very visible beneath the pointy armor. In other words, if your bosses can't be convinced by the brilliance of what you have planned, try to reach a middle ground where they can feel like they've contributed but the image is still reflected in the design.

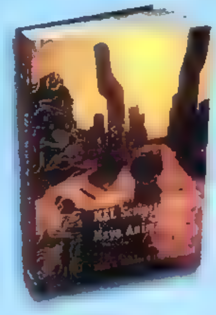
MORGAN KAUFMANN PUBLISHERS
An Imprint of Elsevier Science

LEVERAGING TECHNOLOGY

Web Bloopers: 60 Common Web Design Mistakes and How to Avoid Them
by Jeff Johnson

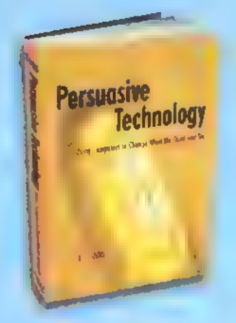


Complete Maya Programming: An Extensive Guide to MEL and the C++ API
by David Gould



MEL Scripting for Maya Animators
by Mark R. Wilkins and Chris Kazmier

Digital Video and HDTV Algorithms and Interfaces
by Charles Poynton



Persuasive Technology: Using Computers to Change What We Think and Do
by B.J. Fogg

Real-Time Shader Programming
by Ron Fosner

Programming Cameras and Pan-Tilts with DirectX and Java
by Ioannis Pavlidis, Vassilios Morellas, and Pete Roeber

Texturing & Modeling: A Procedural Approach, Third Edition
by David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley

JIM BLINN'S CORNER

- Dirty Pixels
- A Trip Down the Graphics Pipeline
- Notation, Notation, Notation

MORE TITLES FROM MORGAN KAUFMANN

- **3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics** *by David H. Eberly*
- **The Art and Science of Digital Compositing** *by Ron Brinkmann*
- **Artificial Intelligence: A New Synthesis** *by Nils J. Nilsson*
- **Advanced RenderMan: Creating CGI for Motion Pictures** *by Anthony A. Apodaca and Larry Gritz*
- **Linkers and Loaders** *by John Levine*
- **Computer Animation: Algorithms and Techniques** *by Rick Parent*
- **Subdivision Methods for Geometric Design: A Constructive Approach** *by Joe Warren and Henrik Weimer*
- **Creative Evolutionary Systems** *by Peter J. Bentley and David W. Corne*

Visit us on the web at: WWW.MKP.COM

ORDERING INFORMATION

Order Fulfillment Department, Elsevier Science, 11830 Westline Industrial Drive, St. Louis, MO 63146-9938
Phone: (800) 545-2522 / (314) 453-7010 [Intl.] • **Fax:** (800) 535-9935 / (314) 453-7095 [Intl.]
Email: custserv.mkp@elsevier.com • **For bulk discounts:** NASpecialSales@elsevier.com

The Concept Image at Work

A concept image can be used to great effect with your art staff. If you have picked an image that exists (such as a painting or photograph), you can just make copies and distribute them, explaining how this single image should be used to inspire their work. As visual artists, we have always found this technique helps focus our efforts. The one potentially dangerous question is whether or not to share the image with producers, publishers, and other decision-makers. We suggest doing this with caution, because the concept image isn't the design, and those who don't work visually might not understand that. It is but the anchor point for the design, and might not appear to have anything to do superficially or intellectually with the actual game you're working on.

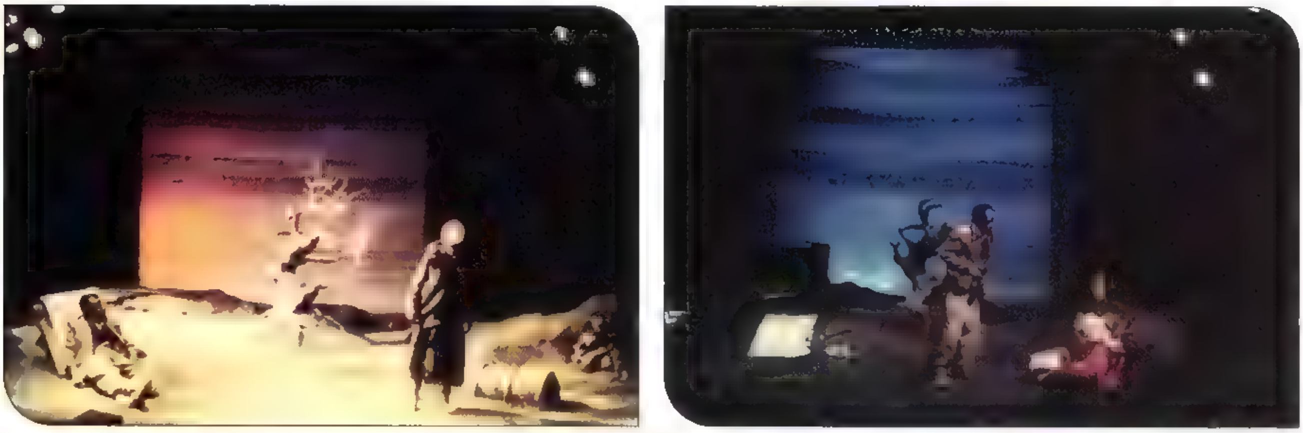
This visual concept functions for an artist like this: Let's say you are a landscape artist. This visual concept is sort of like saying that you intend your next painting to be limited to a narrow palette ranging from violet to blue-green. The values will run from light to dark, but the chroma will be limited to that narrow range. This choice allows you to focus and perfect the other issues at hand: the subject, the composition, and so on. To give another example, say you're going to do a painting with only complementary colors. The point is to use the image you've chosen to help you sift what belongs in the game and what does not. Art is, after all, about making choices. All too often games tend to be compilations of every good idea anyone on the team ever came up with. Back in our *Star Wars* example, someone might present to you a proposed sketch for Luke's and Darth Vader's costumes. Using a very simplistic example, if Luke's costume is dark and angular while Vader's is light and flowing, it's clear that these costumes are in direct opposition with the concept image.

Your image or concept must support the core emotional heart of your game, whether it comes from your own imagination or is chosen from existing images. It should be a very personal image, one that evokes emotion in you. The most important thing is that you actively make a choice, and not simply leave it in your imagination. The more you understand why you chose this image, the more you'll be able to access your own emotional state during development and use the image as a benchmark against which all work will be measured. The art direction of Westwood's EARTH & BEYOND, which Chris worked on, called for a bright color palette, despite some conventional wisdom that futuristic sci-fi games should be dark and gritty. Putting our emotional selves into the color palette selection, however, we believed that mankind's future isn't dark and gritty but bright and hopeful. A different development group might view mankind's future in a different way, and their decisions would be just as valid. The crucial thing is just to decide, and to use your emotions as part of that decision.

Physically passing your concept image around to your art staff is one way to unify design efforts, but in fact you don't have to tell anyone about the image. If you have approval over artists' work or have the kind of relationship with them where they run work past you during all stages of development for your feedback, you can get away with just being there and using the image in your head to guide their creative process. For example, when you critique their early sketches, you could simply communicate that their direction is "too colorful" or "too angular," thus gently nudging them in the right direction. Most staff artists working in the game industry, however, would prefer and benefit from the direction that a concrete image would provide them.

The single-concept technique is especially useful if some of the artwork is

Feeling let down by your 3D software?



Scenes from a production of Boesman and Lena, designed by Chris Klug. A crocus trapped in a broken piece of amber and a broken Coca-Cola bottle served as inspiration for all aspects of set design and lighting.

being developed off-site with a contractor. Assuming the image is strong and clear, sharing the image with the contractor can be like having a clone of yourself on-site with them.

All the World's a Stage, Real or Virtual

Let's illustrate how Chris used this technique in a production he designed. The play was *Boesman and Lena* by Athol Fugard, produced by the New Jersey Theater Forum in November 1978. The main characters are migrant workers in South Africa, living on the edge of existence. Although they love and support each other, life has hardened them to the point where it is very difficult for them to show their softer side to each other. Life has treated these two very harshly. Chris's emotional heart for the play was, "In an environment where everything hurts, love can still flourish." The visual image was a crocus trapped in broken piece of amber and shards of a Coca-Cola bottle. The colors of the glass were used in the light, the sets, and the costumes (see photos). The crocus was used for the color and angle of the moonlight, which represented to the characters their hopes for the future, as well as relief from the unrelenting glare of the daytime sun.

New York Times Arts reporter Robin Pogrebin recently dissected the work of John Lee Beatty, a brilliant Broadway set designer as he was being inducted into the Theater Hall of Fame ("Lush, Plush Or Seedy: Sets Filled With Power," January 21, 2003). At one point this past fall, Beatty had six shows running on Broadway simultaneously, an astounding achievement. Beatty is known for his interiors, and he has described himself as the designer of "the sofa and the staircase." What could be more mundane and repetitive? We all know art staff who would bristle at doing that kind of design, right? But Beatty understands the secret of using visual images and themes to get at the emotional heart of the piece, and uses it to bring his work to a higher level.

We strongly recommend reading the entire article, but snippets reveal Beatty's use of the technique we're talking about. Beatty's sets are "full of character, because they are so much about the people who inhabit them," says director Daniel Sullivan, as quoted by the *Times*. The article cites how the *Dinner at Eight* interior "has its own subversive elements, like the dining table elegantly appointed with silver and stemware that remains suspended in darkness at the top of each act and is never used. 'Your average realistic designer wouldn't think to have that hanging like a guillotine above these ultimately doomed people,' said André Bishop, artistic director of Lincoln Center Theater. 'It's a brilliant abstraction.'" (You can probably guess something that might have been Beatty's visual image for this design.)

The article continues: "Beatty has visual themes running through his productions. There are no windows in the set for *Dinner at Eight* because he said he wanted to convey a sense of 'closed off worlds, people who have cocooned, the way people live in Manhattan.' Similarly, he used black in every scene of a recent production of *Tartuffe* so that 'like the script, there's a little bit of nasty information' throughout. 'I am like an actor, an interpretive artist,' he said. 'I express emotion through scenery.'"

Our virtual worlds deserve the same kind of emotional commitment. As games become more expensive and time consuming to make, publishers will want to entice audiences who have never played games to give them a try. These new audiences are used to having their scenery designed by artists like John Beatty. Art directors in the game industry are going to compete directly against designers of this caliber. Will we be ready? ✐

FOR MORE INFORMATION

Robert Edmund Jones. The Dramatic Imagination. New York: Theatre Arts Books, 1987.

Robin Pogrebin. "Lush, Plush Or Seedy: Sets Filled With Power." The New York Times (January 21, 2003): p. E1. Full article available from searchable archives at www.nytimes.com.

Power. Master Every Angle. **XSI** v. **3.0**

"To maintain our leadership position in the market, it's paramount that we partner with companies that provide **innovative** and **productive** tools.

Softimage set **the standard** in the games industry with SOFTIMAGE|3D and have done so again with **SOFTIMAGE|XSI** which we have chosen, after evaluating all of the software on the market today, to use as our **main game authoring solution.**"

Yoshiki Okamoto
Executive director
Capcom

Download the SOFTIMAGE|XSI EXP v.3.0 **now** at **www.softimage.com/EXP**

SOFTIMAGE|XSI

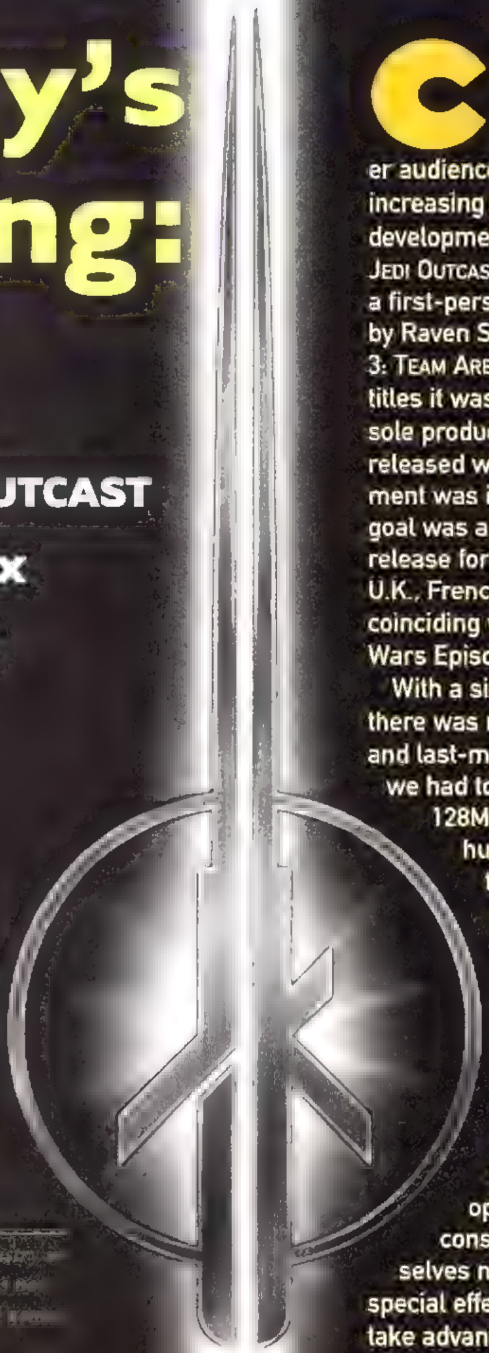
3-D Non-linear Production Environment

Avid

tobi saulnier, bret dunham,

The Play's The Thing:

Converting Star Wars Jedi Knight II: Jedi Outcast from PC to Xbox and Gamecube



TOBI SAULNIER | A struggling duck farmer, Tobi spends her days and nights in the [illegible] of product development for Vicarious Visions, occasionally finding time to [illegible]

BRET DUNHAM | Bret started in the games industry in 1999 [illegible]

KARTHIK BALA | [illegible] visions [illegible] childhood, in his spare time [illegible]

JEZ SHERLOCK | An aspiring goat herder, Jez has been a professional game developer [illegible] currently he herds [illegible]

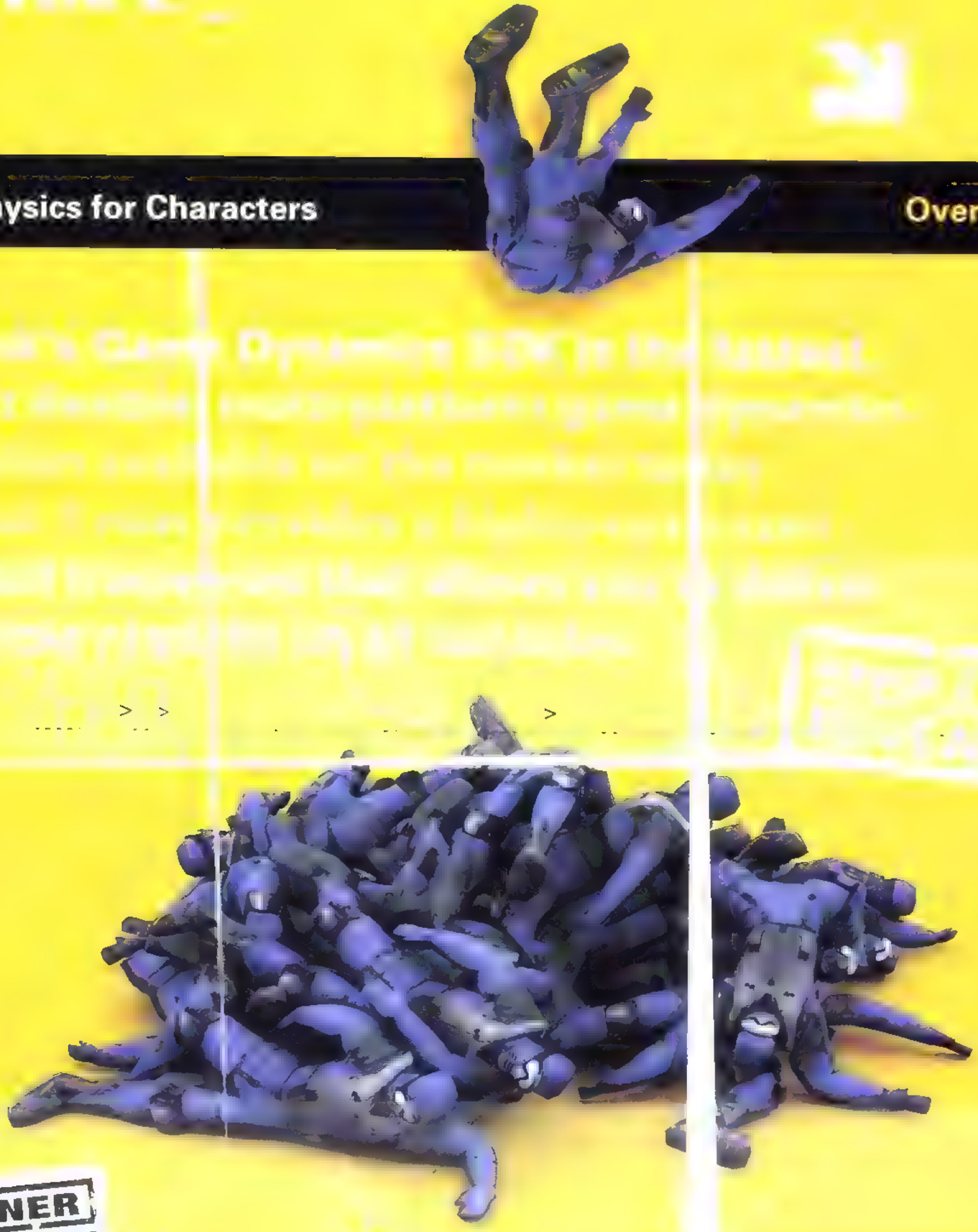
Cross-platform PC and console development is becoming more common as a way to reach a broader audience and help underwrite the increasing production costs of game development. Star Wars Jedi Knight II: Jedi Outcast used to be confined to PC, a first-person shooter (FPS) developed by Raven Software based on the Quake 3: Team Arena engine. Like many PC titles it was never planned to be a console product — in fact, the PC title was released when the console development was in the planning phases. The goal was a simultaneous worldwide release for Xbox and Gamecube (U.S., U.K., French, and German versions), coinciding with the release of the Star Wars Episode II DVD.

With a six-month development cycle there was no shortage of late nights and last-minute inspiration. Somehow we had to cram a PC game with a 128MB RAM requirement, with hundreds of megabytes of textures, onto the Gamecube (with 24MB main memory and 16MB ARAM) and Xbox (with a relatively generous 64MB). The graphics engine had to be rewritten for both Xbox and Gamecube, and the assets needed to be converted and optimized to fit within console constraints. The graphics themselves needed to be enhanced with special effects and detail that would take advantage of the specialized capabilities of each platform. But all of these efforts would have come to nothing unless the game played as well, if not better, on the console. Going in, our mandate was that it had to be a fun console experience, and not feel like a substandard straight port of a PC game.

Havok 2

Physics for Characters

Overview



Havok's Game Dynamics SDK is the fastest, most flexible, multi-platform game dynamics solution available on the market today. Havok 2 now provides a [illegible] framework that allows you to deliver [illegible] on all platforms.

WINNER
GAME COMPONENTS
GameDeveloper
FRONTLINE
AWARD '02

Havok was recently awarded "Best Game Component" in the 2002 Game Developer Magazine Frontline Awards. This award was decided by the users of all middleware solutions in 2002. Havok was judged on earlier versions of the Havok SDK, giving us the opportunity to enhance Havok 2 even further prior to launch.

(see what Rob Heubner of Nihilistic had to say inside)

Want to see how fast and scalable Havok 2 really is? Need exact performance metrics? Check out www.havok.com/havok2 for details

havok

With Havok 2 you get

A focus on game-physics _

The Havok team has one focus – building game specific solutions that work. That focus has made Havok the No.1 provider of middleware physics to the games industry today.

It's bullet time all the time with these ragdolls

Character dynamics

Humans and monsters can now react naturally in physical environments, saving you from having to animate multiple different reactions.

The Havok 2 Character Kit features:

> **Classic ragdoll effects** and **new localized ragdoll effects** blended with keyframed animations, for more realistic death scenes, spectacular secondary motion and damage modelling. For example, you can get a 'dead' arm if you get shot in the shoulder

> Ragdoll effects for **multiple character types**, we support whatever number of limbs, tentacles or antenae your imagination demands

> Breakable constraints for **detachable limbs**, so you can get as gory as you like

> A **fully integrated, "Quake-style" character controller**, based on boxcasts for optimal behavior and performance

> **Tools for ragdoll set-up** for 3ds max™ and Maya®, so that you can easily create, tweak and test your characters in your scene within a familiar environment.

Introducing zombie style ragdoll and keyframed animation blending in Havok 2

Vehicle dynamics _

The Havok Vehicle Kit is designed to allow all types of vehicle behaviors to be simulated effectively.

The Vehicle Kit features:

> **Core vehicle physics system** and **tuned car wheel friction model**

> **Game specific modules** for transmission, engine, suspension, gameplay, aerodynamics and steering

> Breakable constraints for **post-crash car debris**

> Support for **multiple vehicle types** including realistic racing cars, motorbikes, cartoon cars, tanks, monster trucks and articulated trucks and trailers

> **Car Tuning Tool** for real-time tweaking of vehicle parameters

> Havok vehicles can be combined with Havok Character Dynamics for realistic **stunt driver responses**

With Havok 2 the Vehicle kit is an excuse to hit more objects than ever before on the PS2

WINNER GAME COMPONENTS Game Developer FRONTLINE AWARD 02

Frontline award _ continued

"When it comes to Middleware, there are easy problems and there are hard problems. Physics is a very hard problem. No two game projects are exactly alike; different platforms, different tools, and different coding standards abound, yet great middleware has to achieve drop-in integration with any project to be a success. Havok has achieved success in this difficult arena by employing excellent engineers and support staff, both in Europe and the US and by constantly improving their product in response to developer feedback. The first few versions were rough – it was a memory hungry system that was focused largely around PC racing game physics. But in the last couple of years, Havok has pushed hard in areas that are important to gaining broader appeal: aggressive memory savings, more attention to character-based games and focusing on the core physics that developers want most. If the trend continues, maybe we'll be able to move physics into the 'easy' column at last." ***Rob Heubner, Nihilistic***

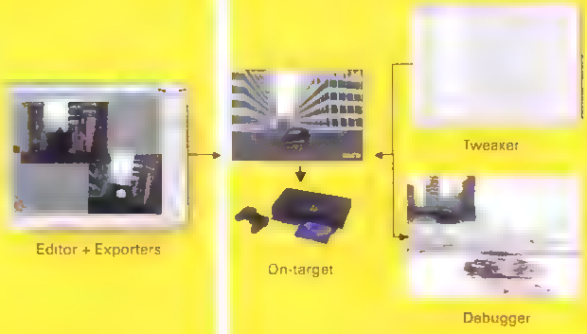
Technical Features

Flexible tools

Havok 2 includes tools for physical content creation, tweaking, debugging and profiling your game. Out of the box, you get:

> **Exporters for 3ds max and Maya** that come with **real-time previewers**, so you can test out your physics set-up quickly and easily. All the Havok functionality is **fully accessible through MAXScript and MEL** to allow easy integration and customization of the exporters

> The **Havok Visual Debugger** provides down-to-the-metal profiling. Use it to analyze your game while it is running on your target platform, identify rogue set ups and profile your game for optimal performance.



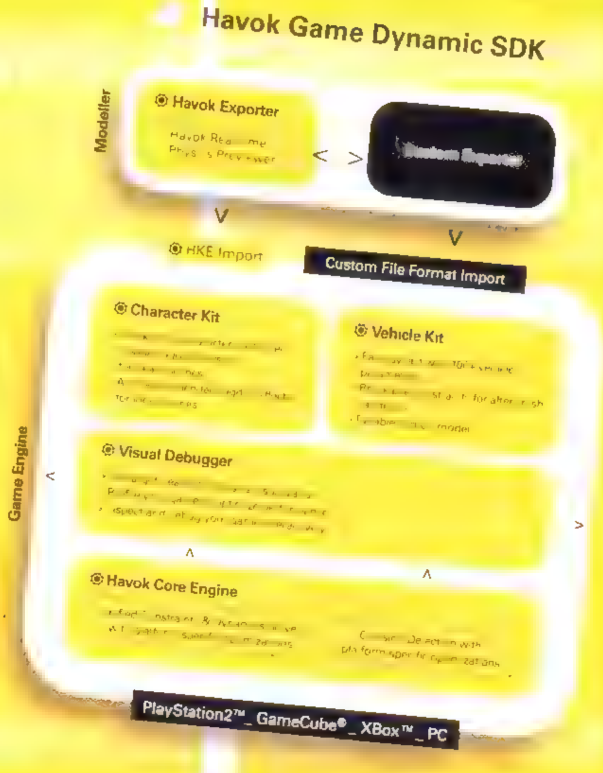
Complete control

> With Havok you have **full control over the simulation accuracy** - you can choose to distribute the physics simulation over multiple frames or use different frequencies for collision detection and core solver to ensure you get the optimal behavior for your game

> The Havok SDK is a **completely modular product** and is **shipped with source**, so you can use what parts you need, when you need them

> You can **extend or even replace** parts of the engine - to get the best out of the integration with your in-house game engine, for example, you can use Havok's collision detection for your AI or graphics components.

Physical interaction between a multi-legged bug, a mechanoid and [illegible]


Havok Game Dynamic SDK
Modeller
Havok Exporter
Havok Real-time Physics Previewer
Custom Exporter
HKE Import
Custom File Format Import
Game Engine
Character Kit
Vehicle Kit
Visual Debugger
Havok Core Engine
Collision Detection with platform specific optimizations
PlayStation2™ _ GameCube® _ XBox™ _ PC

Multi-platform support

Havok is available for the PlayStation2™, Nintendo GameCube®, Xbox™ and PC. It is optimized for each platform, in particular the PlayStation2™ (the engine fully exploits VU0 and uses hand-coded assembly for many core algorithms). The same API is available for all platforms, making it trivial to port your game physics.

Havok can be used with in-house graphics engines as well all third-party graphics middleware solutions, including Intrinsic Alchemy®, NDL Gamebryo®, Renderware Graphics® and the Unreal™ and Quake Engines™.

Client Listings

Publishers

Activision	Eidos	LucasArts	Ubi Soft	Sony
Crave	Empire	Microsoft	Valusoft	Take 2
Dreamcatcher	Infogrammes	Midway	Vivendi Sierra	THQ

Developers

1 Up	Edge	Lodestone	Nihilistic	Sojourn
Candella	Furious	Microids	PCP&L	Sproing
Core	Gaume	Microsoft Games	Paradox	Team 17
Cyan	Grin	Minds Eye	People Can Fly	Valve
Data Design	Ion Storm	Mucky Foot	Planet Moon	Vicarious
Digital Anvil	Legha	Nd Devi	Remedy	Vision

Client Testimonials

Dave Brickhill, Director of Technology, Activision
"Havok's technology is beyond reproach. We use it in several projects and have seen it evolve into a tool both robust and easy to use. It's now far beyond what people can do themselves. As far as I can see, Havok has no competition in the physics middleware space.

Markus Stein, Technology Lead Programmer, Remedy Entertainment
"Since we started using Havok physics, their support has been very well organized and professional and they have helped the Remedy team a lot during the physics engine integration. We have achieved a high degree of customization in a short time-frame while implementing widespread collision detection and character controllers for the game. We are very happy with the performance too, and the integration of new features is generally easy. We are confident that Havok will save us lots of time and money further down the road. So far we are only scratching its very surface.

Jay Stelly, Project Leader, Valve
For useful design you'll need to be able to easily set up forces, collision events, springs and constraints. It's also critical to be able to design objects that start up in a sleeping state and convert from sleeping to active in an easy fashion. Controllers or actions need a pretty easy interface, as this will be where a lot of game play will end up being created."

Bill Baldwin, Project Director, Brute Force
Digital Anvil and Havok physics have [illegible] realism and gameplay into both upcoming titles, Brute Force and Freelancer. The results will be apparent to developers and end users alike. Havok took care of the basic collision detection and response, and then we pushed the envelope using the constraint system for more complex physical situations like character ragdoll physics.

Warren Spector, Executive Producer, Ion Storm
"Havok is a powerful, off-the-shelf, multi-platform tool that really meets our needs. You don't often hear developers "oohing" and "aahing" but even the dev-team members were blown away when they first ran through a map after we got the Havok code running.

company

havok

The Console Experience

What characterizes a console game versus a PC game? If you're a game developer, you probably immediately think implementation: how to fit all that PC goodness into the run time memory, storage space, and polygon-crunching constraints of console. If you're a consumer who's juggling a library of games at home, you probably start thinking of installation details and minimum specs, where you're going to play the game (TV versus computer station), and the genres characteristic of each. In this article we focus on the main boundary between the two camps — the game experience itself. Console games play differently from PC games.

The console experience is defined by two dominant factors: the controller and the player demographic. So even though we might have the same game design, assets, and underlying game code as the PC version, the players' experience should and in fact needs to be fundamentally different depending on what platform they are playing on. Even with all the technical hurdles, the most difficult issue in moving a PC title to a console platform is playability.

The Console Controller: A Blunt Instrument

The main difference between PC and console controls center on the console's lack of a mouse and keyboard. For a first person shooter like Jedi Knight II, the mouse keyboard combination provides a rich array of options for what is assumed to be a fairly sophisticated player, one who is willing to read instructions and memorize more than a few keystrokes. This traditional reliance on a complex input scheme has traditionally made FPSes hard to get right on consoles. The importance of finding good mapping functions from mouse and keyboard to the 14 to 18 buttons of a controller shouldn't be underestimated. But we found that determining a suitable mapping scheme only got us part way there. Even more critical was introducing automated player assistance such as auto-aim and auto level to compensate for the relatively coarse control on a console. In cases where even automated player assistance wasn't enough, we even went as far as making level and AI modifications.

Map This!

Mapping controls from PC to console involves two separate issues. One is just the obvious smaller number of buttons on the controller. The other is the physical differences in how these buttons are accessed and manipulated by the player. In the end we found that the key to finding good solutions to both of these issues is to focus on what the player needs to manage. One of the standards we used for comparison was Halo, which we saw as opening the door for FPSes on console. Halo does a great job of simplifying what the player needs to manage. With about 10 action buttons, the developers were able to keep the learning curve low and the gameplay fast. Having the wealth of PC buttons available actually can work against this goal of simplicity, since it is so easy just to add more keys. In the end, this complexity makes it harder for the player to learn and memorize all of the different options, even on the PC.

Back to the numbers problem. What jumps out at anyone spending a few seconds thinking about it, is that the controller for a console has many fewer buttons (14 to 18) than the PC mouse and keyboard combination (approximately 56 bindable keys for Jedi Knight II). We had to find the best way to translate the huge variety of input variations used in the PC title to the handful available on console. This meant some hard choices, and plenty of opportunity for debate. Controller mapping is a very tangible issue, so we weren't surprised that everyone had an opinion.

Something worth noting is that for a conversion from PC to console, we weren't starting from a blank slate. Unlike a new console game, where control mapping can also be a hot topic, doing a port actually caused more difficulty than a new design. For one thing, when at all possible it was preferred that the same conventions be used as the PC title. For instance, one idea to simplify player tasks was to eliminate the secondary fire for weapons (for instance, the E11 rifle has a primary fire that is a single shot and a secondary fire, independently triggered, that is a burst mode), and rebalance weapons accordingly. Certainly there are pros and cons to this solution, but that was viewed as diverging too far from the PC version.

[illegible]

CPU Pentium II 350MHz
(PIII recommended)
128MB RAM
OpenGL 16MB 3D accelerator
(GeForce, Radeon, etc.)
665MB hard drive space for installation,
additional space for swap file
and save games
16-bit sound card

[illegible]

[illegible] Xbox: 64MB RAM; Gamecube:
24MB main RAM plus 16MB ARAM
No HD on Gamecube. HD use on Xbox only
for save games
Small Gamecube memory card: 512K
Need fast loading time off disc
DirectX 8 rendering engine on Xbox,
Gamecube-specific API for rendering
on Gamecube (not OpenGL)
Console controller for input, not mouse
and keyboard
Controller configuration varies with platform
(number and placement of buttons)

Working from a PC title adds the disadvantage of having a huge number of preexisting functions and mappings, each of which someone has already grown to love. It's important to keep in mind that we as game developers and PC game players probably appreciate the variety of customization and input options more than the typical console player. Furthermore, the typical console player generally doesn't have the patience to wade through a manual and take time learning the mapping. They expect the controls to be easy to learn through simple trial-and-error.

Our approach was to identify which functions were redundant or repeated in the PC interface, and then simplify the rest. We didn't use a highly scientific process for this — it was more a matter of prototyping and testing to evaluate different options as we went. We got ongoing feedback and requests for changes from player- and QA-testing at the publisher and licensor, and we also heavily relied on internal folks who were big FPS fans. This latter group became important in tuning the game to make sure it was a similar level of difficulty on PC and console.

We also looked for conventions with which players would be familiar. Since established conventions used in console platfomers are of little use for a FPS, we studied other console FPS games to see what other developers had done right and where they went wrong.

An example of a function we just removed completely was the snap-to-center view function. On console this was felt to be disorienting. Instead we used a more subtle "auto-level" to shift the camera back to center view as the player moved forward. More about this later.

An example of combining functions was the lightsaber style selection button also being used as the lightsaber selection.

We got most of the way in the numbers game just in rethinking the select function. Compared to a typical console game, JEDI OUTCAST has a huge number of items the player needs to select between: weapons, items, and force powers. These are all mapped to different keys on the PC, whereas on the controller we essentially had the D-pad. We selected the D-pad because in an FPS the digital control is ill suited to either moving or aiming.

Our first design involved using the up and down buttons to toggle between three "tumblers" on the HUD, representing a row for each of the three categories, weapons, force powers, and items. Then, once the desired category was on-screen, the left and right button could be used to toggle the active selection to the next or previous weapon, force power, or item. Although this seemed intuitive to the development team, even that degree of overloading buttons proved too confusing to players. The context of the left and right buttons depended on what category had been selected, and in the heat of battle players did not want to have to remember which context they were in.

The solution was a scheme where the Up button always mapped to Next Force Power, the Down button mapped to Next Item, and the Left and Right buttons allowed Next and Previous weapon selections. Although this meant that the player needed to traverse through the whole list of force powers or items if they overshot, the advantage was that they always got a consistent response from the D-pad buttons. Although this particular solution worked well for the first part of the game, where quick weapon selection is needed, it was not as well suited to the second part of the game when the focus turns to force powers.

Control Freak

Complicating the mapping itself is the fact that not all controllers are the same (Figures 1 and 2). As for any cross-platform title it was important for us to find a control scheme that would translate across different controllers. Things to consider included the finger layout and some of the subtle differences between buttons. We had to tweak the controller sensitivity differently for Xbox versus Gamecube in order to get a similar feel on both. Similarly, the shoulder buttons required specific calibration. For instance, the Xbox controller has only two trigger inputs, compared with the Gamecube's three. We didn't spend much time with the Xbox "S" controller, mostly due to lack of time.

Although designing for the lowest common denominator can be frustrating at times, we found unexpected advantages

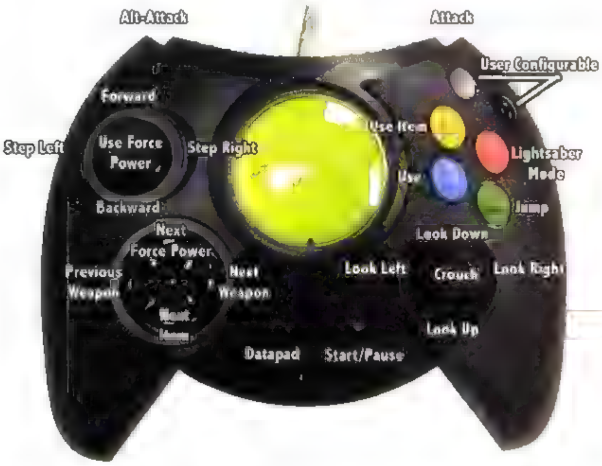


FIGURE 1. Xbox controller input scheme

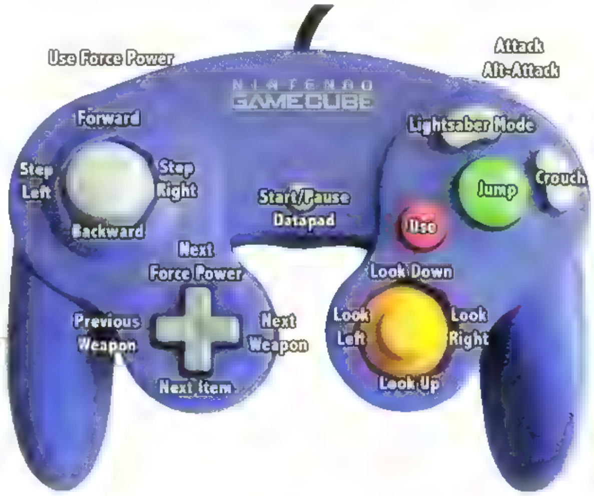


FIGURE 2 Gamecube controller input scheme

GRAB A WILDCAT!

Wildcat

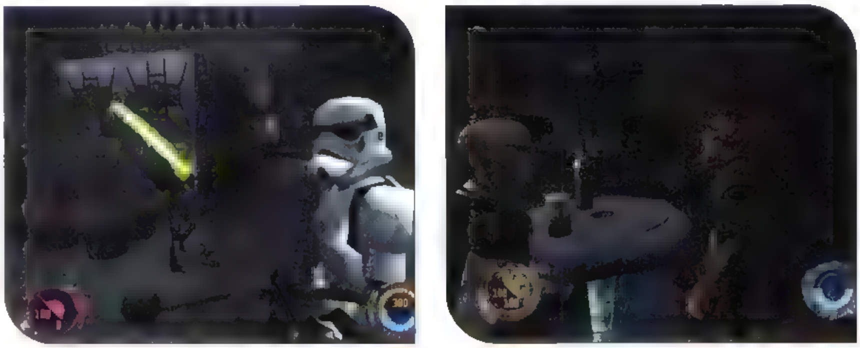
through trying to find a simple, intuitive interface. In our case, because the Gamecube controller has three fewer buttons than the Xbox, solving the control mapping was more difficult. However, because of this necessity we discovered simpler mappings for functions. In the end we actually didn't think we needed more buttons, since the number of things we had to manage already was formidable.

The Xbox controller has three more buttons than the Gamecube, and we're not about to let any go idle. We bridged the gap between the two controllers by saving one key on the core mapping by using a context-sensitive "Use" button on Gamecube (the "B" button). The use-item-in-the-world function (such as opening a door) and use-item-on-yourself function (such as a health canister) were combined into one context-sensitive Use button. For example, if the player was next to a door, a "Use" icon would appear on the door, signifying that the Use button would activate the door. Although we could have applied this function to both controllers, it was confusing for some players; in the final stages we left it in only for Gamecube (which needed the extra button savings).

That left the Xbox black and white buttons free to provide an oft requested feature — the ability to map to the player's favorite weapon, force, or item. Mapping is dynamic and can be reset b simply holding down the hot swap key for a few seconds. Although this was a nice-to-have feature, it was not critical to gameplay, so not a priority for fitting on the Gamecube. Player testing will help us better understand whether that customization is utilized by console players.

You Want Me to Do What?

Aside from the numbers, we have to consider basic ergonomics when choosing what button to assign to what function. Unlike a keyboard-mouse combination, with a controller the player does not have a lot of flexibility in how to reach keys that are needed. Many combinations or sequences of keys have to be ruled out just from a physical standpoint, in terms of finger reach and flexibility. This was a constant struggle with Jedi Outcast because the control is complex — the gameplay requires three attack buttons, one each for primary fire, secondary fire, and force power. So the player may need to move, turn, and fire either the primary or secondary mode, while also using a force power. (Easy for a Jedi, maybe.) Maintaining comfortable access to three attack buttons on the controller became the dominant design constraint, just because of the number of fingers required. A redesign of weapons to eliminate the need for secondary fire might be one future solution.



Screenshots from Star Wars Jedi Knight II: Jedi Outcast. The one on the left is taken from the Xbox version of the game, the one on the right from the Gamecube version.

Beyond the main game mechanic, our overall strategy here wasn't exactly rocket science: buttons close to the player's fingertips were used for frequently needed functions, while those infrequently used were assigned further away. This required playing the game and learning it well enough to determine what a player was going to need to do more frequently. As we learned the game better, our button mappings were refined.

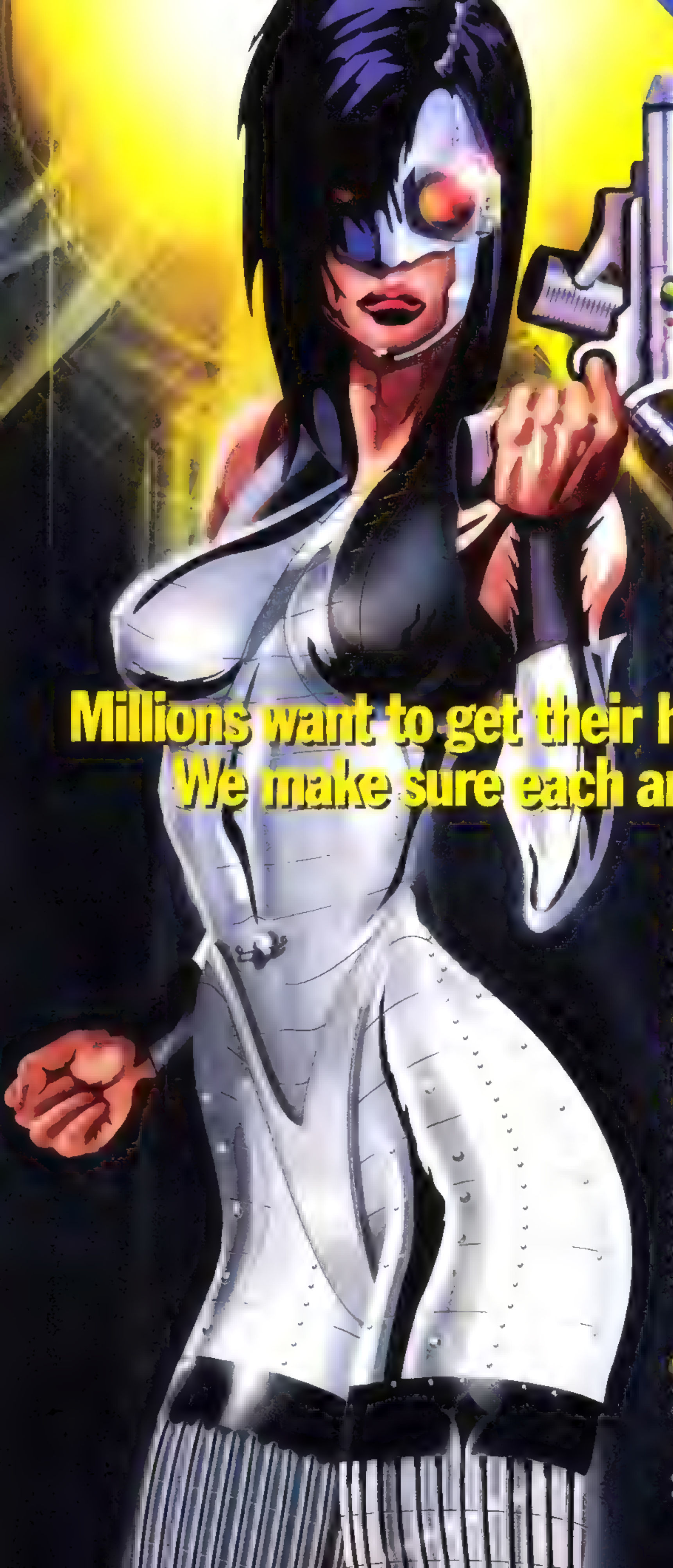
We went through several rounds of testing controller mappings. When testing them we paid careful attention to what players accidentally hit or caused to happen.

A Sticky Situation

Once you have determined the ideal controller map, the next gameplay obstacle is the controller response, in particular the controller sticks. Controller thumb-sticks do not allow for the degree of precision that a mouse on a PC provides. The limited discrete settings for a thumb-stick just can't compare to the typical DPI resolution of a typical mouse. So no matter your players' skill, they will never have the same fine-tuned aiming control as they would have on PC. In addition, the human thumb does not have the same precision of control in the forward backward axis as it does side-to-side, further limiting vertical control.

And then we have the different control paradigm of a controller stick. Navigation via controller sticks can be disorienting to players. For a PC mouse, when the player stops moving the mouse, his or her movement on screen stops. For a controller stick the tilt of the stick determines the movement, so that if the player wants to stop he or she needs to return the stick to the neutral position. At best, this characteristic introduces some inertia in the control, as the player has to wait for the stick to snap to neutral when he or she wants to stop input. At worst it can be counterintuitive, since unlike normal eye-hand coordination, movement on screen is no longer directly related to hand movement. In games with frequent direction changes and the need to scan the environment, the player can be left confused about which way to turn the stick to get the desired effect.

When working out a solution for these inherent limitations, we kept in mind that a player needs to be able to have precise aiming when facing a target, and



REGISTER TO WIN
A FREE TRIP FOR 2 TO
E3 EXPO 2003

[illegible]

You build your games to be unique, entertaining, and above all—successful. Stop unauthorized licensing and duplication from cutting into your bottom line. HASP® and Privilege™ from Aladdin Knowledge Systems offer the strongest yet easiest solutions for protecting your IP, from physical to electronic distribution. Give your hot properties the protection they deserve.

Millions want to get their hands on her. We make sure each and every one pays.

Plug into HASP4®

- Protect your beta distribution with HASP4 Time—a real-time clock gives you full control of your beta's use.
- Prevent illegal copying of your arcade games—control up to 112 games with one HASP4 Memory key.

Check out Privilege™

- The choice of top-tier software publishers for comprehensive electronic software distribution and product activation.
- Feature rich: anti-piracy protection, multi-tier distribution, financial transaction handling, localized in 16 languages, and more.

Secure your game at the highest level.
Call 1-800-562-2543 or visit eAladdin.com to learn more about maximizing your profitability through better anti-piracy protection.

Aladdin
SECURING THE GLOBAL VILLAGE

eAladdin.com

quick 180-degree turning in other situations, such as turning to face an enemy behind the player. Our motto when designing the controls was "Give the player what he wants, when he needs it." An immersive experience requires that the player not have to struggle consciously with the control sticks. We wanted to assist the player as subtly as possible, to keep the game challenging but not frustrating. After studying other PC games and the few FPS games on console, and experimenting with alternatives, we found three solutions that greatly simplify the learning curve for players: sensitivity adjustment, auto-level, and auto-aim.

Sensitivity Zone

Sensitivity adjustment as used in PC games tunes the input device (mouse) input rate, which has a dots-per-inch to pixel ratio. Most PC FPS games allow the user to adjust this so that the game either responds more quickly or less quickly to the user. For console games we need to accomplish a similar adjustment for the thumb-stick input.

We had identified the problem that the player had to be able to have very fine-tuned control when aiming, yet use the same stick to quickly turn around to face an enemy. Our solution was to use a zone-based input to be able to provide both ends of the control spectrum to the player. Zone-based input is simple — the response is adjusted for very precise turning when low input is used and ramped up to radical turning when full input is used. Although this is a classic control scheme, we had to use an ad hoc process to find the numbers that "felt" right based on a lot of experimentation and player testing. For instance we started with a linear mapping of numbers that was way too sluggish for turning. We also started with an eight-zone system (think of zones as concentric ovals emanating from the thumbstick) but found three zones worked just as well (and was simpler for tweaking).

Zone	Sensitivity (pixels per second)
Inner	Decrease
Middle	One
Outer	Increase

TABLE 1. Three-zone input scheme for console controller thumb-sticks.

We figured out that we could cheat some by taking into account what the player was doing and then dynamically adjusting the numbers. For instance if the targeting reticule is activated (colored red) then we assume the player is trying to aim and lower the overall sensitivity.

The final system was a three zone system where:

- Input while in the inner zone (barely moved from center) lowered, approximately by 50 percent.
- Input in the middle zone was not scaled.
- Input in the outer zone (thumbstick pushed way over in some direction) increased after a delay, approximately by 45 percent.

The time delay buffers the input in case the player accidentally flicks the thumb-stick to the extreme outer edge. Interestingly, we had to use different values for the time delay for Xbox (500 ms) and Gamecube (250 ms), due to the different thumb-stick sensitivity. The modified stick input level was combined with the player-set sensitivity (via the menu) and converted to degrees per second movement rate.

The final change we made to the control sensitivity was to have different sensitivities for the X- and Y-axis movement. Due to thumb movement limitations, players have difficulty looking up or down quickly. This occurs when players overcompensate for the lack of flexibility and end up pushing the stick in a diagonal direction. To minimize this accidental movement, we made the X-axis response greater than Y. Given the natural motion range of the thumb, controller sticks are much harder to control in the Y-axis versus the X-axis, so this modification was intended to compensate for that effect.

Aim to Please

Effective player assistance should be subtle enough that players never feel we are doing the aiming for them. Players want and need to know they were skillful enough to overcome the game's challenges and enemies, so allowing all the shots to hit every time is a mistake because it becomes obvious theyre being helped. We learned this first-hand when our early prototypes used too-accurate aiming, and while it was momentarily thrilling to be able to hit everything, it quickly became boring. In this case there was already a hit/miss record built into the game, so we took the ad hoc approach of having a few players at different skill levels play both the PC and console versions and tuned the auto-aim so that the ratio was approximately the same. This approach assumed that the PC game balance was good.

The auto-aim implementation itself was simply a matter of adjusting the projectile trajectory sufficiently to allow a higher chance of striking the target at which the player is aiming. In particular, once the aiming reticule changes to red, indicating that an enemy is in the player's sights, we lowered the horizontal sensitivity, but not the vertical. This gave the player enough additional fine control to overcome any aiming problems caused by the thumb-sticks, yet allowed for a small percentage of shots to miss. The benefit is that the player can turn quickly, from victim to victim, and more easily lock on toa character for a kill. The ammunition's trajectory could not go outside the reticule in order to hit the target, which would have made it obvious to players that they were getting a helping hand. It is important to maintain the illusion and feeling of controlling fire in the gameplay experience.

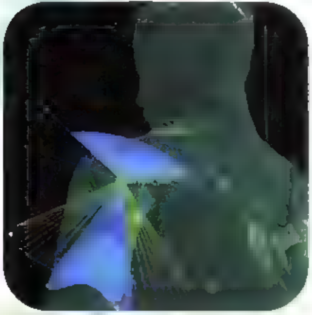
The remaining weakness of this approach is that AI enemies that move quickly from side to side are harder to hit, even with auto-aiming, because by the time the fire reaches where they were, they have moved. This has always been the bane of auto aiming. Rather than try to implement predictions of where the AI would moving to, we

KARMA

Performance and control

KAT

The Karma Authoring Tool allows you to visually edit joint limits, collision volumes and every other aspect of your simulation. Its multiple window approach allows editing in one window while the simulation runs live in the other. Even complex cases such as blending animation and dynamics are catered for.



Karma is the leading phy[illegible] middleware from Mat[illegible]

As well as supplying [illegible] the world's most res[illegible] game developers, [illegible] the maker of the dynamics technology [illegible] the Unreal Engine and the forthcoming RenderWare Physics.

Karma is used in over 80 titles and offers the most flexible and optimized solution available for physics in games.

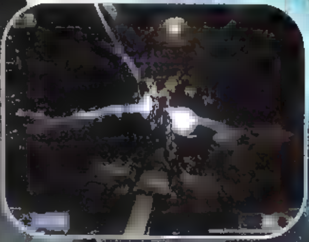
Karma simulates all types of [illegible] dynamics efficiently from single body physics, [illegible] to the most realistic characters in games.

Karma characters feature joints with [illegible] soft and twist limits to give the highest possible [illegible] but the solvers used are scalable allowing developers to simplify the simulation and limit memory and [illegible] usage if multiple characters are required.

More developers than ever are choosing Karma to provide fast, realistic, multi-platform physics that's suitable in every genre of game.

"The Math Engine guys contacted us and said, "Hey, we think we can improve your physics dramatically with Karma." and we gave them the source code. Six weeks later they came back with a really impressive demo and they showed us all of this stuff they had been talking about and it really worked well in Unreal. Now we're using it in all upcoming Unreal games -Unreal Tournament 2003, Unreal Championship and Unreal II."

Tim Sweeney CTO Epic Games



"Lionhead Studios has chosen to use MathEngine for its games due to its cross-platform compatibilities and its efficient and robust performance. MathEngine has enabled us to quickly and painlessly integrate physics solutions across multiple platforms without the need to rethink each solution repeatedly, and the robustness and flexibility offered by MathEngine lets us spend our time on new ways of actually using physics to enhance the experience of our games."

Peter Molyneux Lionhead Studios

"[illegible] reputation developing [illegible] agreed deadlines. We use Karma's dynamics and collision system for its flexibility, stability, and ease of integration with our own proprietary physics technology"

Brian Jobling - Managing Director - Eutechnyx

RenderWare Physics

Unreal TECHNOLOGY

For further information on Karma please email karma@mathengine.com

mathengine

For further information on Renderware Physics please email rw-info@csl.com

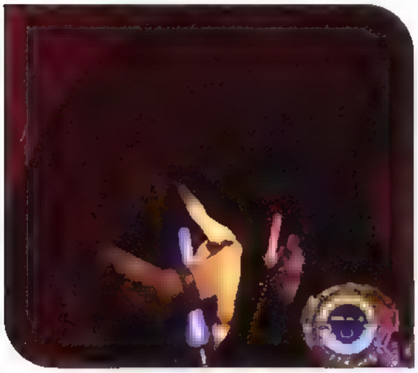
For further information on Unreal Technology visit www.epicgames.com cheated a bit and just slowed down the AI side-to-side movement. Overall we think this made for more manageable AI anyway, since in the PC version they were very fast, probably better tuned to the response time of a mouse.

We also made exceptions to auto-aim, such as when the player used a sniper gun, for obvious reasons. The sensitivity was still lowered to allow a more accurate aim, but the trajectory was not adjusted. An example of an auto-aim feature considered but not selected was an aim mode where targets are cycled through. Although suitable for non-FPS games where the player has limited aiming control, in JEDI OUTCAST if you had to stop long enough to use an aim mode, you'd be dead. JEDI OUTCAST is definitely a game where you need to move and aim and fire at the same time.

Level with Me

The auto-level feature automatically recenters the camera as the player moves forward, to help the player read just the camera to get the view they need at that moment. This avoids requiring them to try to adjust the camera back towards center while also moving, and potentially firing a weapon and so on. This function could become annoying if it happened at the wrong time, for instance while strafing or when the targeting reticule is active (that is, when targeting an enemy), so this is another example of a feature that needs to use some inference about what the player is doing in order to be effective.

Auto-level is nothing new — it is used by almost all FPSes in one form or another. However, a main difference in our implementation of it for the console experience was to leave it on by default, whereas most PC games would have it off by default. We decided a more capable player could turn it off via the menu system if they wanted more precise control. This decision was based on the expected demographic being younger and less skilled than the PC player.



No matter which console the game is played on, the user can still execute moves like this

A Balancing Act

Although control issues are the easiest way to screw up a console game, and particularly a port of a PC title, there are other seemingly minor details that can make a huge difference in making sure that a game plays like a true console game rather than a PC game.

First, the more accurate control provided by a mouse-keyboard combination means that the overall speed and difficulty of the PC game is typically too high for a player dealing with a console controller. One example of how we adjusted the difficulty to match the controller was to slow down the enemy AI, as mentioned previously, so enemies don't move faster laterally than a player can target with a controller.

Another example of playability tuning for consoles is to actually redesign levels and AI placement to minimize the amount of up-and-down looking required of the player, which is awkward on the thumb-stick. For instance, it's wise to eliminate extreme vertical changes in the level, where a player needs to look up and down more than 45 degrees to engage enemies or solve puzzles. Where the PC player with a mouse would have moderate difficulty in such a situation, it is just frustrating for the console player. Adjusting the levels so that they require mostly horizontal aiming is much more satisfying to the console player. In our case we had limited ability to change the level geometry beyond what was needed to fit the assets on Gamecube, however we accomplished the same result by removing AI in those elevations.

One area that worked really well on console was the use of the thumb stick for third-person lightsaber mode. Perhaps because the attitude and physical control of a lightsaber is similar to a thumb-stick, this was the one area where it was actually easier to control with the console controller. A lightsaber is an ideal case for controllers not only in the types of movements the weapon uses, but also in that a high degree of precision is not needed to be deadly.

Conclusion

Looking back on the process of converting STAR WARS JEDI KNIGHT II: JEDI KNIGHT OUTCAST from PC to consoles, we can credit much of our success to our approach of planning parallel efforts in some areas and then choosing which worked best. We did this mainly because with a six-month development cycle we didn't permit for a learning curve, but in the end it made us a lot more efficient. Software folks will recognize this as the spiral lifecycle of development, where rapid prototypes are used to retire biggest risk areas first. We avoided rushing down just one path, which saved our bacon a couple of times when we hit unexpected dead ends. Many of these were technical in nature — creative approaches to squeezing the assets down to console size that just couldn't be evaluated any other way but trying them on the assets. Amazingly, we got it done on time, and hit every milestone on plan.

Game reviews and feedback are saying that our console version plays well and it has a good "console feel" to it, which obviously was not part of the original PC title. Needless to say, this experience has led to even more ideas to try out next time. As the console audience continues to mature, we can expect to see a continued trend toward FPS games on console. We hope to be involved earlier in the process next time, so that decisions made during the original game development can take into account impact on both PC and console playability.

Lost Toys' BATTLE ENGINE AQUILA

BATTLE ENGINE AQUILA, Lost Toys' second game, was built around the desire to create a "next-generation" shoot-'em-up game combining the core playability of titles such as 1942 and RADIANT SILVERGUN with cutting-edge technology and graphics. The concept behind BATTLE ENGINE AQUILA's take on the genre was simple — instead of following the current trend of having a lone player battle against incredible odds, why not re-create those epic action scenes seen in films such as *Starship Troopers* and *Saving Private Ryan*? The player is not a lone soldier, but part of a vast fighting force — albeit one which is doomed to fail without the support of the players' vehicle, the Battle Engine Aquila.

From the outset, it was clear that BATTLE ENGINE AQUILA would be an ambitious project. The original design called for massive battles between hundreds of individual units, all controlled by their own AI and able to react to anything the player was doing (not by extensive scripting). The player would have the freedom to roam throughout the battlefield and interact with everything that was happening. Effectively, we would be creating a complete, large-scale RTS game engine, and then attempting to provide FPS-style gameplay and graphics on top.

MOHO (released in the U.S. as BALL BREAKERS), Lost Toys' first title, had been a relatively small project, [illegible] production on BATTLE ENGINE [illegible] team was expanded to allow for the [illegible] process that continued [illegible] later stages of development.

Even though we were targeting consoles (specifically the Xbox and Playstation 2) for the final game, all of the development work was done on the PC. It wasn't until late in development that we moved the code base over to the two consoles, and even then it was only the programmers and testers working on those consoles that ran the game on them — all the artists and designers used the PC version of the game. This turned out to be both a blessing and a curse.

What Went Right

1. Flexible core technologies. From the project's [illegible]

BEN CARTER | *Ben has been workin[illegible]e games industry since 1995, both as a freelance journalist for magazines incl[illegible] Play, Edge, and G4, and was a lead programmer on two titles for the Acorn RIS[illegible]form (MERP and MIRROR IMAGE), and one on the PC (ABSOLUTE TERROR). He has also been involved in writing games coverage for many nonspecialist press publications such as Manga Max and The Irish Times. Having spent the last two years working on PS2 and Xbox engine code for BATTLE ENGINE AQUILA, he is now working on the graphics engine for Lost Toys' as-yet unannounced future projects and spending far too much time watching anime. Contact him at ben@sailune.net.*

GAME DATA

[illegible]

(PC version forthcoming

[illegible]

and about 500 individual game objects

As the designers and artists became familiar with the various editors and scripting systems, they were able to create effects and missions that we had previously assumed would require custom code. For example, for a level requiring players to chase after a retreating enemy battle ship, we had been contemplating writing a complex system to enable the game's normally static (and constrained) world map to scroll. While the programming team was attempting to figure out how to achieve this, the level designers implemented the mission without it by misusing some of the scripting functionality in a clever way. This was an unexpected side effect of the system's flexibility.

It was only near the end of the project that we were forced to scale back on this flexible approach (reasons for which are explained further under What Went Wrong) and implement more optimized specific routines for certain areas of the game. Even then, however, the engine's modular construction frequently allowed faster special-case code to be substituted for the generic routines without actually altering the interfaces, and without any code outside the module being modified.

2. Constant play-testing and feedback. Since all the development — except that of the actual console versions of BATTLE ENGINE AQUILA — was done on the PC, everyone on the team could play the game at any time on their own machine, without needing to borrow a development or test kit and a TV. This made a huge difference to the development process, as team members not directly involved with the programming and design could see their work in the game almost instantly — a model could be exported from 3DS Max straight into the format and location needed. The artist responsible could then run the game and see changes immediately.

This functionality also helped with the game's sound effect and music creation, which was handled out-of-house. By providing the audio contractor with a copy of the PC–development build and our custom sound effect editing and placement tool, the contractors could experiment with different effects within the game, and then send us a complete set of sounds and the effects file (which mapped game events to specific sound files, and allowed alteration of relative volume, pitch-shifting, and the like). We could then drop these files straight into the game with no need for format conversion, file renaming, or other such annoyances.

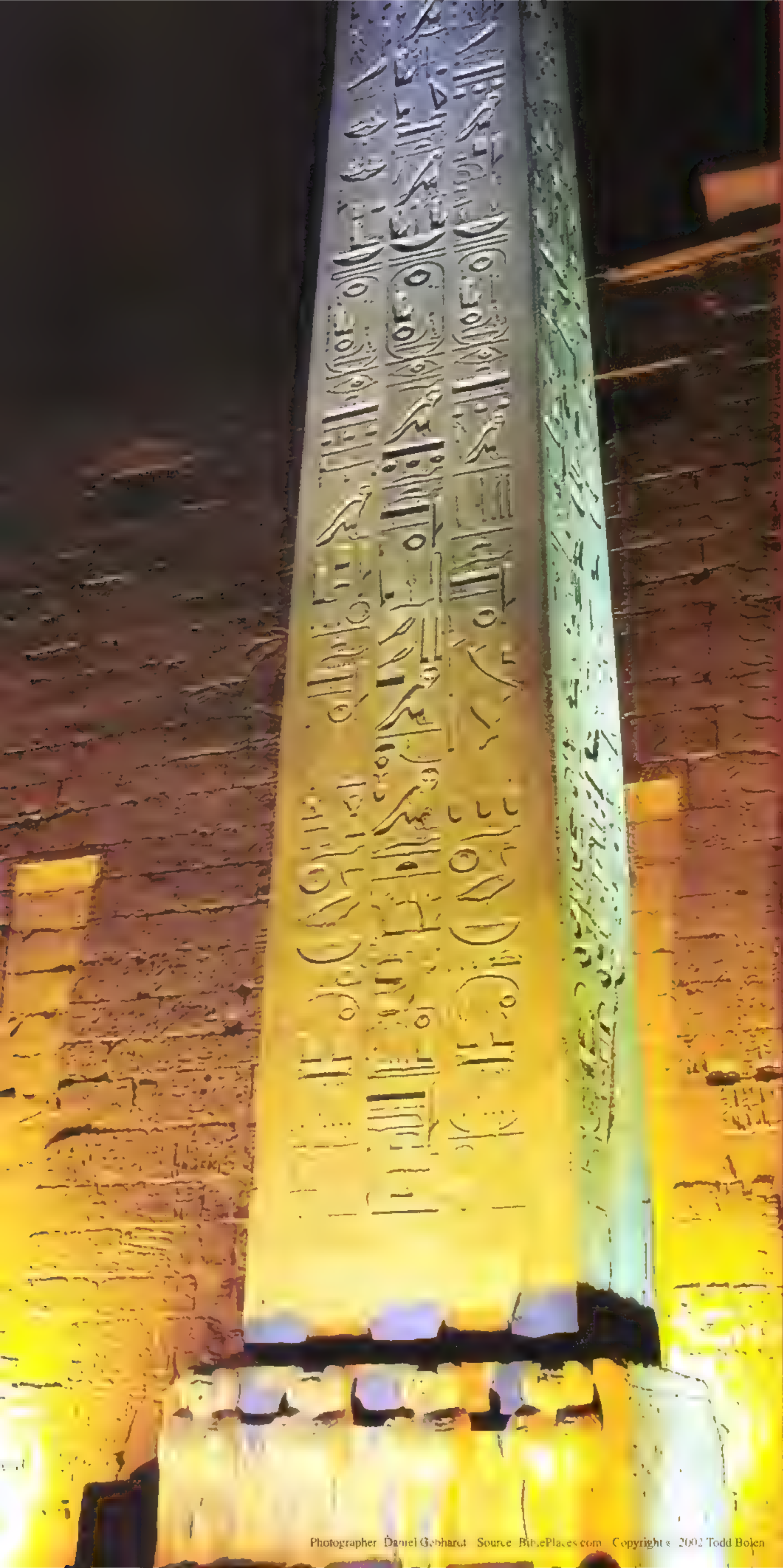
The final and probably most beneficial effect of constant play-testing was a steady stream of invaluable feedback to the design team on gameplay and level-balancing issues. This feedback allowed us to assess the impact of changes to parameters such as the Battle Engine's handling and available flight energy, and to catch those cases where minor changes caused major differences in certain players' game experiences.

We were also able to catch a lot of bugs with this process. It was not uncommon during development for changes to artwork or scene units to break certain levels (buildings being placed close together and then intersecting each other when the model changed, for example). But with many pairs of eyes constantly inspecting the whole game, these mishaps were generally found — and fixed — quickly.

3. Planning localization and porting in advance. We knew from the beginning that BATTLE ENGINE AQUILA was going to be a console game, and that it would probably end up being translated into many different languages for international export. Hence, by planning for localization and porting issues as early on in development as was practical, we attempted to avoid as many problems as possible.

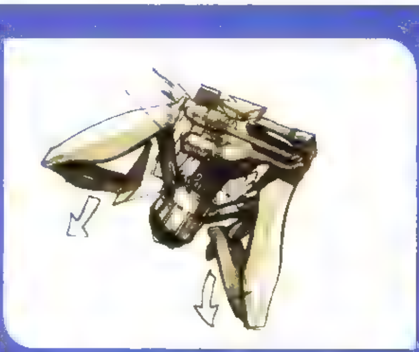
The majority of the engine was structured in such a way that platform-specific code was collected into small modules, which were then called upon by the higher level platform-independent code. By keeping code separated out like this, we were able to remove the PC implementation of these modules and insert Xbox and Playstation 2 replacement code with relative ease — in theory at least, the porting process consisted of taking all these blocks of code and replacing the blank function calls with the appropriate code for each platform.

While we did encounter some porting problems (described later), in general the approach worked very well. Until we deliberately split the code bases for final tweaking and testing, the game could be built on all three supported platforms from one set of project files. This was a huge boon, as the majority of development in many areas of the game (gameplay, scripting, AI, and so on) was performed entirely in platform-independent code, so no effort was

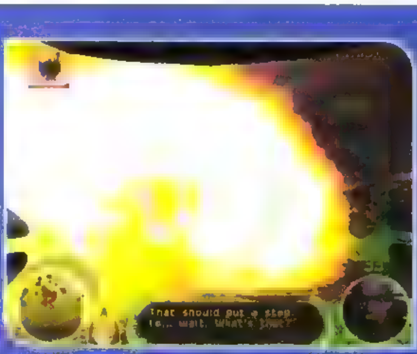


Photographer Daniel Gebhardt Source BiblePlaces.com Copyright © 2002 Todd Bolen

An early version of the Battle Engine in flight, one of a number of discarded designs.



Most game models were composed of a large number of independently animated parts.



BATTLE ENGINE AQUILA's explosions and other effects were created without programming assistance by designers using our custom particle editor.

required from the programmers on each platform to ensure the latest changes were integrated and working.

To simplify the effort of translating the game text and dialogue into the five languages we supported in the final version, we developed a text management system that split all the text out from the code and scripts and instead allowed individual strings to be referenced by a special tag. For example, FRONTEND_NOMEMCARD would translate to a string informing the user that there was no memory card inserted. These tags could also be used to reference the appropriate speech sample for a spoken version of the text, if it was available. Later, we extended this to allow platform-specific variations on tags as well as language-specific ones, so that we could have different sets of messages for the two consoles where their naming conventions differed.

After a few initial teething problems, this localization system worked beautifully. It enabled us to provide the translation teams with a single file containing all the text required for the game, and then reintegrate their localized versions with relative ease. Some careful file management allowed later changes to the dialogue and messages to be separated out for retranslation, and in the final stages of testing we also implemented a simple text viewer that could display all of the messages in the in-game language file so that the localization QA teams could easily check that their work had been incorporated correctly.

4. Ambitious goals. It was obvious from the original design document that some of the technologies we were setting out to create were going to require an enormous amount of work. We wanted a game with hundreds of units, huge walking robots, detailed terrain, fully destructible buildings, and realistic physics. We wanted the player to be able to experience these features at every level, from alongside the tiny troopers swarming around the battlefield to high in the sky looking down across the entire conflict. Even with the overwhelming scale of the initial design, we still considered outlandish ideas for additions ("The islands should have dense forests on them that the walkers and tanks can flatten paths through as they go," "You should be able to land on large enemy craft and destroy them from the inside," or "If you blow up buildings, there should still be rubble when you return to that island later") as potential challenges, rather than dismissing them as technically too difficult or time-consuming.

This mindset caused us more than a few headaches later in development when we realized just how complex and resource-hungry the game had gotten, but the dedication of everyone working on the game ensured we could include a huge number of these ideas that were not originally planned. If we hadn't aimed so high and constantly attempted to achieve the impossible, the game would not have been as fun or technically impressive.

5. Open atmosphere and good communication. It's something of a cliché in these Postmortems to mention the value of good teamwork, so I won't dwell on it, but that doesn't make it any less of an important point. The entire Lost Toys team worked well together on the BATTLE ENGINE AQUILA project, and a lot of potential problems were averted by having relevant people talk them over beforehand.

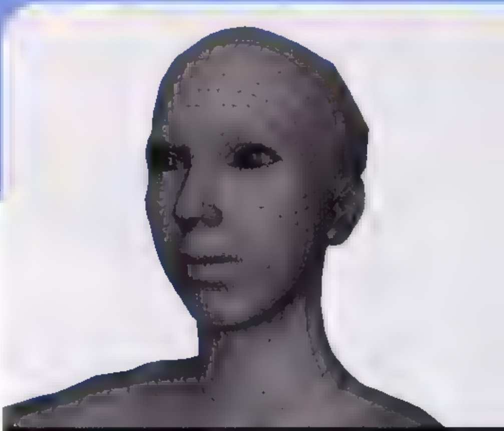
With the entire company based in one open plan office, it's always possible to walk over and ask questions. This way we are able to pool expertise from everyone involved. It's often the case that even if someone isn't working directly on a given aspect of the project, that person has some relevant knowledge that can be helpful.

What Went Wrong

1. Late console development. The vast majority of the development work on BATTLE ENGINE AQUILA was performed on PCs, and it was only about nine months before the game went gold that we finally started working on real development systems for our two target platforms, the Xbox and Playstation 2. Although the PC-centric devel-

3D Scanning At Your Convenience

Eyetronics offers 3D scanning services designed to meet the unique needs of game developers. Our service is fast, easy, convenient and proven in major games and films. We can scan and model human heads, body parts and full bodies, as well as objects big and small.



Features

- **On-location scanning:** Eyetronics does its scanning on the site of your choice. We can do a complete scan of a human or object in about 30 minutes or less
- **Relaxed scanning:** Scanning with our portable ShapeCam is like a photo shoot. Your models don't need to stay in a fixed position for a long period of time
- **High-rez texture:** Eyetronics captures both a high-resolution texture and a 3D mesh in about a second.
- **Model generation:** Eyetronics uses its proprietary software to create custom, textured models to your specifications.
- **Easy import:** Scans can be easily imported into your 3D modeling software through all commonly used file formats (OBJ, 3DS, LWO, XSI, etc.)

Courtesy of Team Soho, SCEE

Inside a 3D scanning session

Eyetronics uses its portable ShapeCam to freely move around the person or object to be scanned. The ShapeCam captures 3D geometry and high-resolution textures by simply taking snapshots with a digital camera. Pictures are taken from different points of view and from various angles. The captured data are processed off-line using ShapeSnatcher Suite software, which includes features for stitching together multiple 3D patches, remeshing (NURBS, subdivided surfaces), cylindrical or spherical texture mapping, and polygon reduction

Eyetronics has provided services to...

Digital Domain/Paramount Pictures ("Star Trek : Nemesis"), Digital Domain/Revolution Studios ("xXx"), MGM ("Bulletproof Monk"), EON Productions/MGM ("Die Another Day"), Paramount British Pictures ("Lara Croft and The Cradle of Life - Tomb Raider 2"), Sony Computer Entertainment Europe ("The Getaway"), Zoic ("Angel", "Buffy, the Vampire Slayer"), Codemasters.

Eyetronics HQ
Kapeldreef 60, 3001 Heverlee
Belgium
Email : info@eyetronics.com
Phone : +32-16-298309
Fax : +32-16-298319

Eyetronics US
811 N. Catalina Ave, Suite 2120
Redondo Beach, CA 90277, USA
Email : info-usa@eyetronics.com
Phone : (800) 205 9808
+1-(310)-937-9702

Eyetronics

opment environment was a great help in some areas, it wasn't long before we realized that working like this for so long had caused some serious bloating of code and resource.

Our code structure was aimed toward making the porting process as painless as possible, but we hadn't counted on the extent of the limitations of the console platforms relative to the PC. It only took us a day in both cases to get the core game engine running on each machine, but there was clearly an awful lot of work left to do.

Fundamentally, the game was too resource-hungry for the machines it was to run on; in terms of memory, the PCs we were using for development had four to 16 times as much RAM as the consoles. In the early days of the porting process, even small levels were regularly using more than 100MB of RAM and running at below 20 frames per second. Right until the game went gold there was a constant battle to get everything to fit into memory, especially on the Playstation 2 where we only had about 28MB of RAM after the game executable had been loaded.

The Xbox port of the game had the advantage of being based on DirectX, and hence the majority of the code was shared with the PC version. The Playstation 2 port, however, required an entire graphics and sound engine to be coded from scratch — a mammoth task for our two Playstation 2 programmers, one of whom had never actually written any code for the machine before this project and was still supporting a significant amount of code on the PC toolchain and Xbox sides of the project.

Thus the Playstation 2 version of the game was playing catch-up with the PC and Xbox versions from day one. While initial development went quickly, a vicious circle developed mid-project, where features were being added to the project faster than they could be ported, and we were still struggling daily to get the code optimized enough to stand any hope of reaching acceptable speeds (or even running at all on retail hardware). It was only in a final burst — after the Xbox version of the game was finished and in final testing — that functionality stopped being added to the engine and we were able to get the port running acceptably.

For the project's last couple of months, most of the programming team was thinking about just two things: how to make the game run fast enough and how to use sufficiently little memory on the two platforms. We used every trick we could think of — structures were ruthlessly compacted, data was decompressed on the fly or streamed off-disk as needed, and on the Playstation 2 we were even forced to store additional game data in the I/O processor's memory and move it into main RAM when it was required.

While some of this effort was a consequence of the ambitious game design, we could doubtlessly have avoided a great deal of pain and effort had we been stricter in working to the limitations of our target platforms earlier in the development process.

2. Too much story, too little script. The story line for BATTLE ENGINE AQUILA went through many revisions before arriving at the version in the final game. Unfortunately, constant editing removed many of the interesting twists, and cutting down on the volume of cutscenes and dialogue (both to keep from overstretching our limited art resources and to avoid bogging the game with irrelevant story) resulted in a faint shadow of what the final plot could have been. This is by no means a major problem, since we intended the gameplay rather than the story line to be the more important factor, but the less exciting cutscenes and characters add little to the game. The time spent creating them would have been put to better use elsewhere.

A more serious side effect of trimming the story was on the mission structure of the game. In making a conscious effort to keep the missions and story tied together, we ended up in a position where we were tied to creating certain missions in a certain order, with little room for maneuver if we felt part of the design wasn't working. While I think we overcame this quite well, one of the main criticisms leveled at BATTLE ENGINE AQUILA is that the missions often have similar objectives with little variation in the settings — a direct consequence of sticking to the structure of the original story line.

3. Poor resource management. BATTLE ENGINE AQUILA's 40-plus levels comprise a bewildering array of files and data, much of which was created by

A typical level can have hundreds of individual troopers, each with independent and squad-level AI our own custom tools before being combined into a set of "resource files," one for each level, which include all of the required data for that mission in a ready-to load processed format. The console versions of the game run exclusively from these combined files, while the PC development version can operate from the raw data for quick turnaround when testing. This system provided us with a lot of flexibility, but it wasn't until close to the project's end that we realized that we'd inadvertently created an unmanageable process for building final output.

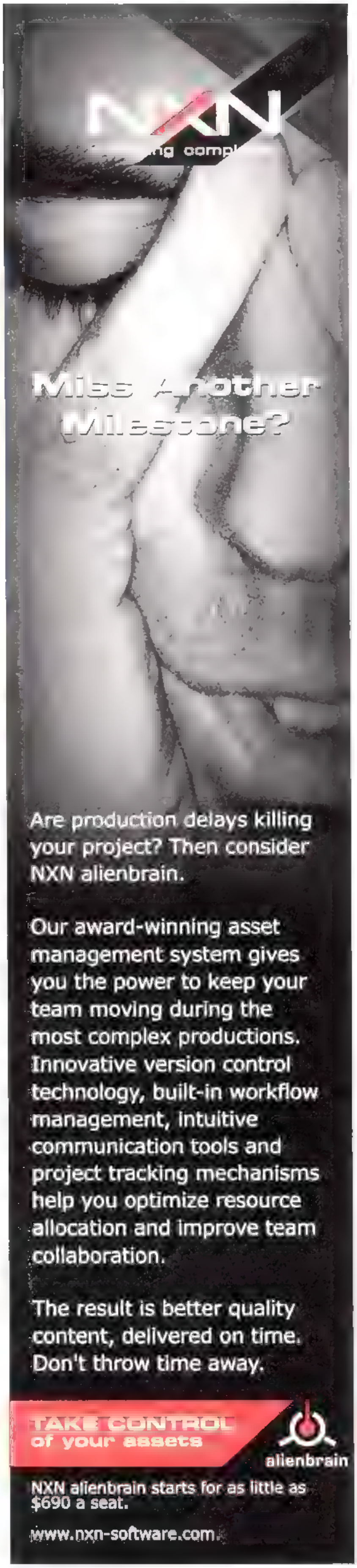
The process of getting a complete build of the game from raw data involved using about five different tools on different sets of data, some of which were only understood by one or two of the team members. There were quirks in many of the tools (such as the level editor saving files with no scripting information unless you had the right set of script files on your hard drive), and we had virtually no version-control system to ensure that the right files were being used. Amazingly, the final game data was a huge directory on the server, which got files dumped into it by all team members.

By far the worst part of this system, however, was the process of creating the final resource files. The PC version of the game ran in a special mode where it would load each level and then dump the contents of its own resource pools into a file, performing operations like compressing textures and precalculating shadow data as it did so. Unfortunately, this process relied on an incredibly risky system of saving objects to disk by writing the entire contents of a C++ class structure and then manually fixing up pointers and other information when it was reloaded. In some senses this worked quite well, as it allowed most additions to game structures to be handled "automatically" by the resource system. But the system caused complete chaos toward the end of the project, as any change to one of the stored structures would render all the existing resource files useless and necessitate a full rebuild of the data — a process that could take several hours for a full level set. It was not uncommon for people to waste hours simply trying to update both their code and resource file sets to be compatible with each other, only to find that in the interim someone else had made another change, rendering the new sets of files useless.

Concept art and production materials were gathered together to form an unlockable "goodies" section for additional replay value

4. Lack of communication between art and programming teams. The similarity between the Xbox and PC versions of the game meant that the screen previews the art team was seeing of their work were almost pixel-perfect representations of what the Xbox version would look like. Unfortunately, this approach to the preview process tended to hide two very important potential problems: performance and the Playstation 2 version.

The artists rarely paid much attention to the frame rate, as the game's speed varied a great deal depending on the specification of the machine it was running on at the time. With no sign-off process for the technical aspects of artwork, it wasn't uncommon for models with ridiculous numbers of textures or polygons to get put into the game.



Problems would only show up when the levels were played on the target hardware, and by that stage it was often hard to tell exactly what was causing the problem. In one case we had an innocuous building mesh that had texture mapping that generated approximately five times as many polygons as the original model contained when converted for rendering. This problem was exacerbated by the fact that until relatively late in development, the programming team was not sure what the eventual limits on polygon counts and texture usage would be, so artists were often given vague or contradictory advice on what to aim for.

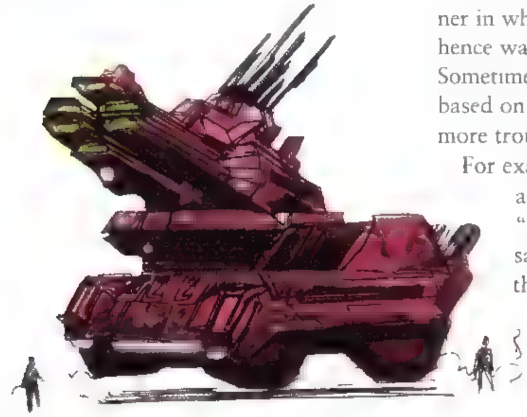
The Playstation 2 version of the game suffered from these problems and more — many of the features supported by the Xbox and PC engines (such as anisotropic texture filtering) were not available, and others had to be turned down or dropped entirely for speed or memory reasons. Until very close to the gold mas ter date, many of the models in the Playstation 2 version of the game did not look right or caused immense performance issues, and it was not until we made a concerted effort to produce and work through a list of problems that we really managed to bring the problem under control. Even then, we found ourselves making modifications to artwork to fix odd problems mere days before the final build was sent off.

5. Too much flexibility. The advantage of the engine's flexibility became a serious liability when we had to feature-lock the game and get it running within our speed and memory budgets. There were both purely technical and design-related problems, some of which could have been avoided if we had planned ahead a little.

One of the main problems we had was with the exceptions. A lot of our attempts to optimize systems went along the lines of "Well, this functionality is only ever used in this way, so let's just hard-code that instead." We'd generally find out at that point (or sometimes only after actually making the code change) that there was one place in the game where this rule didn't hold. There would be one particle effect that used a certain awkward blend mode, or one type of unit that had a nonstandard friction setting. In some instances we could change the errant case so that the optimization would still work, in others we were forced to abandon the optimization or code in yet another special case to handle the one-off situations.

A similar issue that crept up a few times was that some of the systems were so flexible that they were being used for things they were never designed to do. While in some cases such uses were perfectly reasonable and even quite clever, in others they posed a major problem. Code was not optimized to work in the manner in which it was being employed, and hence was running very inefficiently. Sometimes further functionality had been based on this behavior, leading to even more trouble when trying to optimize it.

For example, trees were originally added to the game as standard "things," handled in much the same way as units, troops, and the like. Trees could be shot at, knocked over, or block line-of-sight, which seemed a neat addition to the game at the time. Unfortunately, we realized a few weeks later that some lev els now had in excess of 6,000 individual tress on them (which accounted for nearly 2MB of RAM at one stage), and re-engineering the code to handle this efficiently without breaking the now-established behavior took a great deal of thought and effort.



The transformation sequence for the Battle Engine uses a complex blend of traditional animation and procedural techniques

Battlefield Stories

Developing BATTLE ENGINE AQUILA was a tough struggle at times, and there are many things we'd undoubtedly do differently if we had a chance to do it all again. Our experiences should allow us to avoid making the same mistakes again in the future, freeing us to discover a host of new ones. But game development wouldn't be the vibrant, ever-evolving field it is without fresh pitfalls to uncover at every turn.

At the time of writing, the game has not yet been unleashed on the public. We're all understandably nervous about how the title we have slaved over for the last two years will be received, but I don't think there's anyone here at Lost Toys who isn't immensely proud of what we have created. We've managed to produce a finished product without compromising the original concept and gameplay that we first aimed for, and the finished BATTLE ENGINE AQUILA is a remarkably accurate reflection of that original vision. That, above commercial success or critical acclaim, is surely the greatest thing a developer can hope for.

Cannonballs!

We Want You

Art
Animators, Level Artists, [illegible]

Design
Level Designers, [illegible]
Designers

Programming
3D Graphics, AI, PS2,
Maya

Send resume and
portfolio/reel to:
[illegible]

Cranky Pants Games
803 Kirkland Avenue
Suite 200, Kirkland, WA
98033

Crankypants™ games
A THQ COMPANY
www.crankypantsgames.com

Are you Ready for Life on the eDGE?

Edge of Reality, Ltd. is a veteran entertainment software developer focused on next generation consoles. Located in beautiful Austin TX, the studio is currently hiring for a major license based project. Edge of Reality is currently looking for talented individuals with a passion for games to join our highly creative team!

eDGE oF rEALITY

Positions Available

Senior Environment Artist	Senior Tools programmer
Senior Character Artist	Effects/Lighting artist
Environment Artist	Senior Texture Artist
Senior Animator	Lead Designer

See our Web site for more details! **www.edgeofreality.com**

tAKING gAMES to the eDGE

ArenaNet

is hiring

3D Character Modelers
2D Texture Artist

Game Programmers
Network Programmers
Sound/Music Programmers

Work with key members of the teams that created **Warcraft Starcraft, Diablo** and **Battle.net** on their next project

Find detail on how to apply at our Website: www.arena.net or mail to
Arena.net 10020 Main St., #164,
Bellevue, WA 98004

TIMEGATE STUDIOS

The studio behind the award winning Kohan franchise is growing! Check out these exciting opportunities:

Art
Texture Artists

Design
Sr. and Jr. Designers

Programming
Lead, 3D Graphics, A.I.

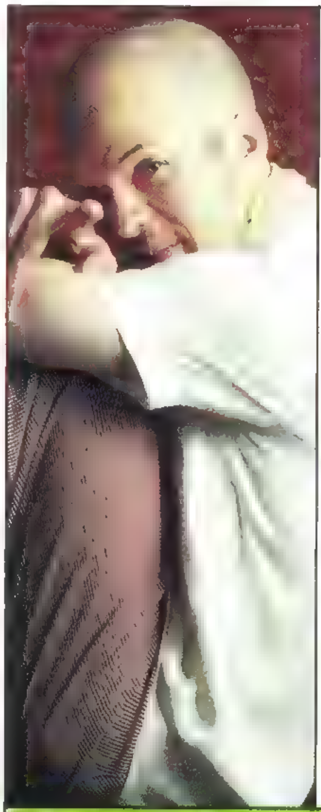
- Competitive salary
- Extensive benefits program
- Dynamic environment
- Stable infrastructure
- Talented team

To apply, send resume and portfolio/demo to:

jobs@timegate.com

TimeGate Studios, Inc.
Attn: Human Resources
14140 Southwest Frwy
Suite 400
Sugar Land, TX 77478

www.timegate.com



To advertise in Creative Careers, contact Raelene Maiben at 415-947-6225 or e-mail rmaiben@cmp.com

To target those looking to Get Educated, contact Aaron Murawski at 415-947-6227 or e-mail amurawski@cmp.com

Enlist Now!

Dynasty Warriors4

Firaxis Games is Hiring!

"★★★★★ Editors Choice"
Computer Gaming World Magazine

"PC Strategy Game of the Year"
- Gamespot
AIAS

"PC Strategy Game of the Year"
- Computer Games Magazine
- GameSpy
- AIAS

"Wargame of the Year"
- Computer Gaming World Magazine
- Computer Games Strategy Plus Magazine
- Gamespot

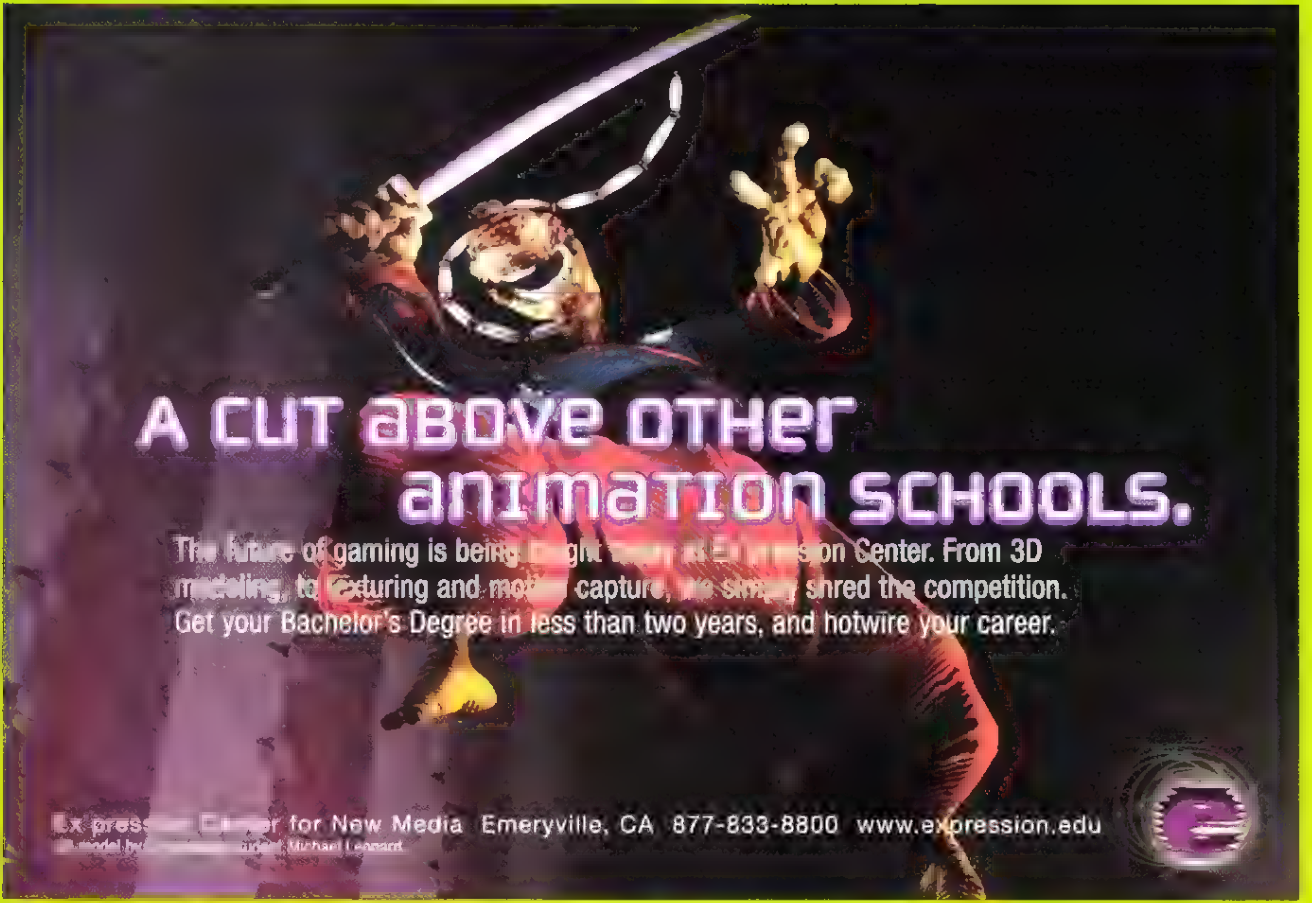
Join the team that's created some of the greatest games of all-time!

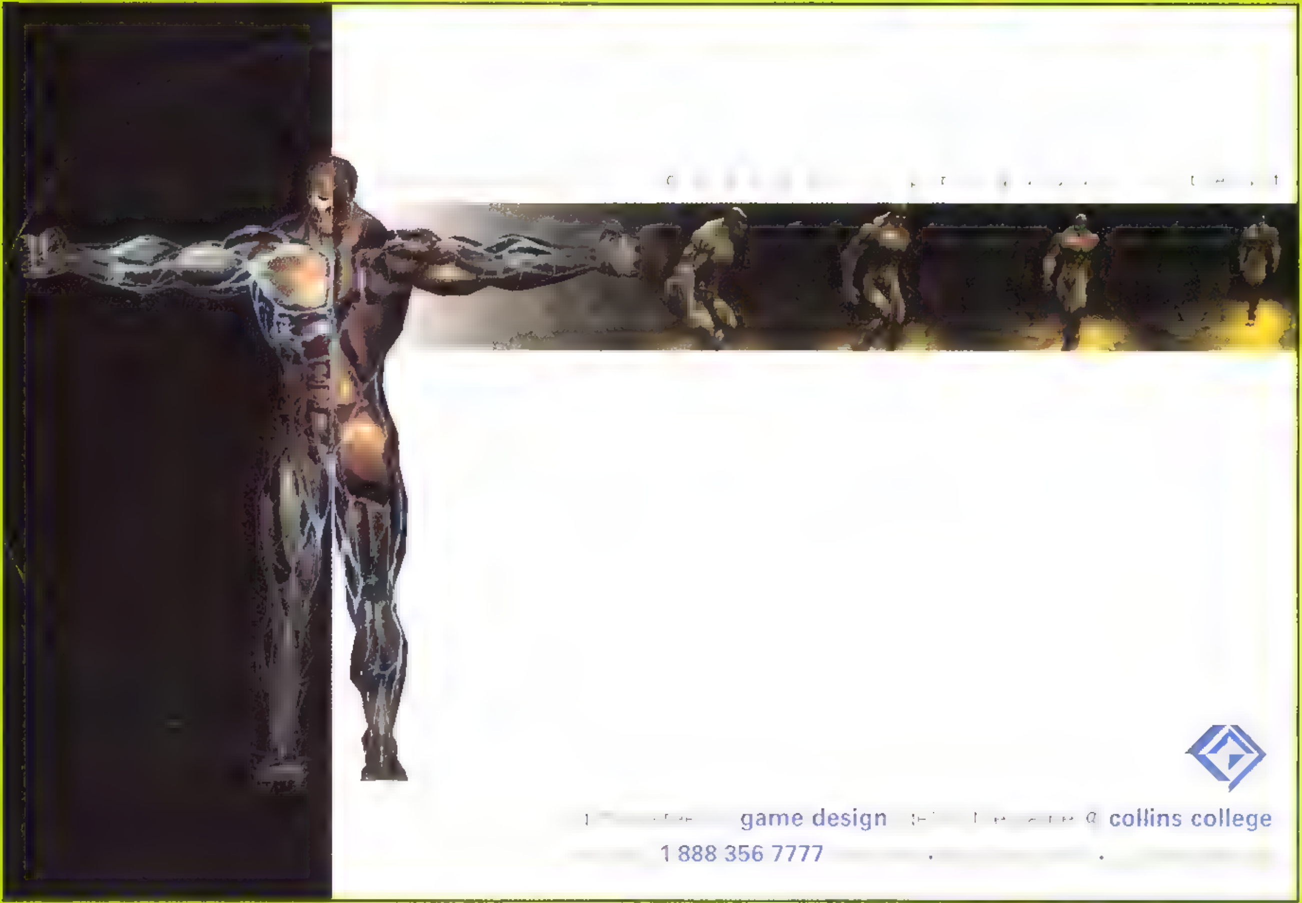
Firaxis Games, home of legendary designer *Sid Meier*, is expanding and looking for seasoned, game industry programmers and artists to help create the next wave of award-winning, best-selling games! Firaxis is an independent development studio that provides a highly creative, casual environment where artists, programmers and designers work side by side to craft cutting edge computer games. We offer competitive salaries and excellent benefits, including matching 401K, profit sharing, generous vacation and more.

For more information or to send your resume please email: **makegreatgames@firaxis.com**

Or visit us at: **www.firaxis.com**

FIRAXIS


A CUT ABOVE OTHER
animation SCHOOLS.



Gowoon Choi, AAC Student

Turn Your Ideas
Into
GAMES

digital world
MESMER PRESS
XSI Illuminated: Foundation
XSI Illuminated: Character
Maya Illuminated: Games
3ds max Illuminated: Foundation
Photoshop Illuminated: For Animators

To target those looking to Get Educated contact Aaron Murawski at 415-947-6227 or e-mail amurawski@cmp.com

To advertise in Creative Careers contact Raelene Maiben at 415-947-6225 or e-mail rmaiben@cmp.com

TARGETPAVILION

Realm Wars
Marble Blast
Metal Drift

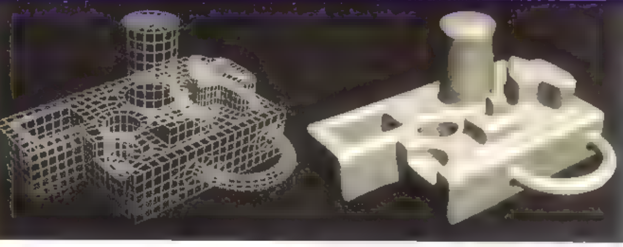
WORLDWIDE PROTEST

BECAUSE EVERYONE'S DOING IT!

THE INDUSTRY UNITES ON MARCH 6 2003

LIQUID DEVELOPMENT. THE ONLY ART RESOURCE YOU NEED.
415 331 3338 WWW.LIQUIDDEVELOPMENT.COM

nPOWER SOFTWARE



GD Conf. Booth # 1147

1) Power Booleans 2.0 - NEW:
   - -- Quad Meshing for Sub-D
   - -- Level Editing User Interface
2) Power Booleans Rhino - NEW
3) Power Solids on Rhino - NEW

POWER BOOLEANS

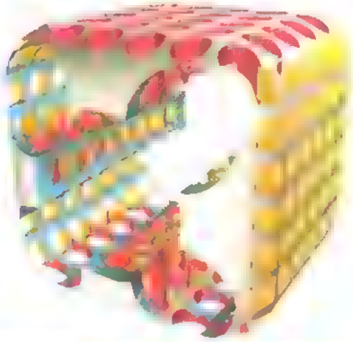
[illegible] MAX Plug-in that provides
[illegible] MAX meshes

Fewer Polygons & Cleaner Meshes.
Fully Animatable Boolean Operations!
Improved Usability with Multi-Object Selection.
Integrated Decimation Polygon Reduction!

POWER SOLIDS

Power Sol ds introduces a trad.t on
[illegible]

View Dependent Tessellation for Rendering.
Import IGES, Rhino,STEP & SAT files
Powerful Construction Primatives.
Filleting, Shelling & Booleans!
Animation of all Solids Operations.



www.npowersoftware.com
(858)-538-3800

continued from page 80, column 1

> The press has to agree not to slam games without doing our homework, and developers have to agree not to slam reviews without considering our criteria.

a game's not worth $50, I'll say so. The only time I ever really think ill of developers is when I hear them complain about reviews they didn't like, and not because the reviewer's point wasn't valid — just because they wanted a perfect review. *GamePro*'s highest rating is a 5.0; I've gotten negative feedback on scores as high as 4.5. Huh? Please accept my constructive criticism for what it is — constructive — and take a step back from your baby.

I have tried programming. I suck at it. I truly do appreciate the blood, sweat, and long nights that you've put forth over the last 18 months. But in the same way that I don't hold the entire industry responsible for one bad game, I hope developers don't lump all the press into one pile of unqualified egomaniacs.

The press has to agree not to slam games without doing our homework, and developers have to agree not to slam reviews without considering our criteria. The press represent your most loyal fans, and we can't wait to see the magicians' next trick.

DAN AMRICH | *Dan is senior editor at* GamePro *magazine.*

continued from page 80 column 2

that there are a lot of other forces at work when it comes to influencing the average game consumer — marketing campaigns, name recognition and licenses, price point, and so on.

Reviews that don't live up to developers' expectations are almost inevitable, since everyone who reviews a game uses slightly different criteria to judge a title, and developers generally don't complain about that.

Given that inherent fact, I really wish that the same reviewers at magazines or web sites would review games within the same genre. When I see publications where reviewers are jumping between FPS games, racing games, RPGs, and everything else, as a consumer I feel like making valid comparisons between ratings is fruitless. And as developers, sometimes we're left wondering, "Wait, this guy just said in his review that he hates platformers. Why is he reviewing our game?"

Ultimately, though, if we're charging people money for our games, they have the right to say whatever they want to about them. It doesn't matter if the comments come from a senior editor or an 8-year-old on a forum. The only time I think we really have a right to complain about bad reviews (outside of the bitching and moaning we do with the rest of the team) is when people get the facts completely wrong, or if there's clear evidence the reviewer barely played the game. And even then, we're probably better off keeping our mouths shut, quietly evaluating all criticism, and working to make the next game an improvement.

TED PRICE | *Ted is president of Insomniac Games.*

ADVERTISER INDEX

COMPANY NAME	PAGE
3Dlabs	43
Academy of Art College	69
Aladdin	45
Alias|Wavefront	C3
AMD	C2
Antrho	9
ArenaNet	63
Art Institute Online	71
Blizzard Entertainment	65
Buglord/Octopi	76
Butterfly.net	21
Charles River Media	22
Cogswell Polytechnical Institute	70
College for Creative Studies	64
Collins College	68
Cranky Pants Games	63
Criterion Software	2–3
Discreet	37
DTS	23
Edge of Reality	63
Exequo	59
Expression Center for New Media	68
Eyetronics	55
Firaxis Games	67
Full Sail Real World Education	70
Game Institute	72
Garage Games	75
Havok	41a
Integrity Ware	77
Koei	66
Liquid Development	77
Math Engine	47
Mesmer Animation Labs	70
Morgan Kaufman	35
New Pencil	75
Nokia	16
Numerical Design	25,26
NXN Software	57
Okino Computer Graphics	22
Omni Interactive Audio	76
Oregon 3D	36
Premier Press	15
Programmers Paradise	5
Qualcomm	7
RAD Game Tools	C4
Red Eye Studios	78
Redshift	53
Savannah College of Art and Design	73
SDL Sheffield	74
Singularity Studios	74
Softimage	39
Testing Testing 1,2,3	11
The Hartcenter at SMU	73
Time Gate Studios	63
Touchdown Entertainment	29
University of Advanced Computer Technology	72
Vancouver Film School	72
Walter & Company	64
Wild Tangent	60–61
Zona Inc.	13

dan amrich & ted price

When Game Developers and Game Reviewers Collide

Illustration by Belle Mellor

Dan's View (the Reviewer):

What do reviewers think of when they think of developers? "Thank heavens these people exist, because otherwise I'd be stuck reviewing pocket calculators." Without developers, there are no games to review. These are the folks who create something out of nothing. Developers make magic.

That said, I think the magicians are often wary of revealing their secrets to the press, afraid that their work is going to be misrepresented by some insulting hack with no sense of responsibility. And while some press members do set bad examples, others aspire to noble goals. In general, I think both developers and reviewers want the same thing: to be known for making worthwhile products. Better still, we can both root for the other side as much as our own.

Just as I don't publish first drafts, I understand a developer's reluctance to show a product before it's ready (but without a handful of early screens and basic info, I can't tell the masses what you're up to). It's tough to send out early builds for preview, because you know how much has yet to be implemented and tweaked (but without doing that, we can't tell our readers why they should start saving their money). And it's really hard to cough up a release candidate when you've gotten 10 hours of sleep in the last two weeks and all you can think of is the stuff that still needs to be triple-checked (but if we don't get the game for review in time for our deadline, we can't capitalize on the all-important exciting newness of your release). The press is necessary, but I would like to believe we are not a necessary evil.

Most developers I've met take their fun seriously, do what they do out of love, and invest a lot of themselves into each product. Hey — me too. As a game critic, I may be in no danger of winning a Pulitzer, but I still maintain a code of ethics. If

continued on page 79, column 1

Ted's View (the Developer):

What do developers think of when they think of the press? How about, "Oh man, are they going to like our game? Please, please, God, let them like it ... Did we get it to them in time? Did anyone remember to activate the nude cheats?"

Waiting to see how your game fared in the enthusiast magazines and web sites is like waiting to see your grades posted at the end of a school semester. You approach the event with a mixture of fear and excitement, hoping that the reviewers saw everything that was great in the game and ignored all of the crash bugs that shipped out with the beta.

But for all the effort developers put into making a good game and cooperating with the media, 99 percent of the games ever released have received at least one or two bad reviews, by which I mean the reviews fall well below what the developer expected.

Bad reviews are disappointing but can provide eye-opening feedback for developers. We take what the press says very seriously, and sometimes it drives the decisions that we make.

The down side of this feedback channel is that sometimes we take what the press says too seriously, erroneously assuming that reviewers represent average consumers. By trying to please reviewers, we potentially alienate the non-hardcore consumer market. And since we here at Insomniac make games that are for broad audiences, this is especially true for us.

One of the big questions we ask ourselves all the time is, "How much do enthusiast reviews matter to consumers?" I've never seen any hard data to answer that question, but generally developers presuppose that the reviews are really important to all gamers. Most of us are hardcore gamers ourselves, so a lot of us make our buying decisions based on enthusiast reviews.

But review scores may not mean as much to the average gamer as we imagine. There are some shining examples of games that were reviewed badly and sold millions (FROGGER on PSX, anyone?) — it doesn't happen all of the time, but it happens. Many more games are reviewed great but sell squat (our own DISRUPTOR being a good example). We've got to remember

continued on page 79, column 2